PROYECTO FIN DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN DISTRIBUIDA DE GESTIÓN DE INVENTARIO PARA DISPOSITIVOS MÓVILES

Autor: Víctor Pacheco Martín

Tutor: Francisco Javier García Blas

Leganés, octubre de 2011

Título: DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN DISTRIBUIDA DE GESTIÓN DE INVENTARIO PARA DISPOSITIVOS MÓVILES.

Autor: Víctor Pacheco Martín.

Director: Francisco Javier García Blas.

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mis padres, por su paciencia y comprensión durante el largo tiempo desde que terminé las asignaturas, y con todo lo que me ha pasado después.

A mi prima Irene, por su esfuerzo con los iconos, aunque, ¡los necesito un poco más grandes!

A mis amigos y la gente que me quiere, por hacerme más llevaderos algunos momentos de desesperación durante el desarrollo de este proyecto.

Y por último, a mi tutor, Francisco Javier García Blas, por ayudarme siempre y readmitirme cual hijo pródigo después de algunos baches.

Resumen

Hoy en día, en la era de Internet y de las comunicaciones, las empresas necesitan que sus procesos sean ágiles y eficientes, y a tan automatizados como sea posible, ya que no existe tiempo que perder. Teniendo en cuenta lo anterior, este proyecto se centra en optimizar uno de los procesos más importantes de cualquier empresa, la gestión de inventario.

En este documento se detalla el análisis, diseño e implementación de una aplicación para dispositivos móviles *Android* que permita realizar de manera remota y sencilla dicho proceso de gestión de inventario, ahorrando tiempo y recursos.

Asimismo, se desarrollará una aplicación servidor que atienda las peticiones desde el cliente a través de servicios Web, y que acceda a la base de datos para insertar o recuperar la información de inventario de manera transparente para el usuario.

Palabras clave: Internet, gestión de inventario, Android, servicios Web, base de datos.

Abstract

Nowadays, in Internet and communications era, every corporation needs its processes to be as agile, efficient, and automated as possible, because there is no time to lose.

Having that in mind, this project is focused in the optimization of one of the most important process in any corporation, the inventory management.

In this document is detailed the analysis, design, and implementation of an *Android* mobile device application that allows to perform, remotely and in a simple way, the inventory management process, saving time and resources to the daily users.

Also, a server application is going to be developed to attend requests from the customer through Web Services, being able to access the database to insert or get the inventory information transparently to the user.

Keywords: Internet, inventory management, Android, Web Services, database.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN 11

1.1 MOTIVACIÓN 11
1.2 OBJETIVOS 12
1.3 ESTRUCTURA DEL DOCUMENTO 13
1.4 GLOSARIO DE TÉRMINOS 14

2. ESTADO DEL ARTE 15

2.1 ANDROID 15
2.2 ECLIPSE 26
2.3 JSON 28
2.4 REST WEB SERVICES 32
2.5 MYSQL 33
2.6 HIBERNATE 34
2.7 JUSTIFICACIÓN DE TECNOLOGÍAS 36

3. ANÁLISIS DEL SISTEMA 39

3.1 DESCRIPCIÓN GENERAL 39
3.2 CASOS DE USO 40
3.3 REQUISITOS DEL SISTEMA 49
3.3.1 REQUISITOS FUNCIONALES 50
3.3.2 REQUISITOS NO FUNCIONALES 55
3.3.3 REQUISITOS DE RESTRICCIÓN 57
3.4 MATRIZ DE TRAZABILIDAD CASOS DE USO – REQUISITOS 59

4. DISEÑO E IMPLEMENTACIÓN 60

4.1 ARQUITECTURA DEL SISTEMA 60
4.2 DISEÑO DETALLADO 65
4.3 DISEÑO DE BASE DE DATOS 82
4.4 DECISIONES DE DISEÑO Y DETALLES DE IMPLEMENTACIÓN 84

5. PRUEBAS 88

5.1 PRUEBAS DE ACEPTACIÓN 88
5.2 MATRIZ DE TRAZABILIDAD REQUISITOS – PRUEBAS 98

6. CONCLUSIONES Y LÍNEAS FUTURAS 99

6.1 CONCLUSIONES 99
6.2 LÍNEAS FUTURAS DE TRABAJO 101

7. PLANIFICACIÓN Y PRESUPUESTO 103

7.1 PLANIFICACIÓN 103
7.2 PRESUPUESTO 105
7.2.1 COSTES DE PERSONAL 105
7.2.2 COSTES DE HARDWARE 106
7.2.3 COSTES DE SOFTWARE 107
7.2.4 COSTE TOTAL 107

8. BIBLIOGRAFÍA 109

ANEXO I: MANUAL DE USUARIO 111

ÍNDICE DE TABLAS

TABLA 1: GLOSARIO DE TÉRMINOS 14
TABLA 2: VERSIONES DEL SISTEMA OPERATIVO ANDROID 16
TABLA 3: VERSIONES DE ECLIPSE 28
TABLA 4: CASO DE USO CU-01 41
TABLA 5: CASO DE USO CU-02 42
TABLA 6: CASO DE USO CU-03 43
TABLA 7: CASO DE USO CU-04 43
TABLA 8: CASO DE USO CU-05 44
TABLA 9: CASO DE USO CU-06 44
TABLA 10: CASO DE USO CU-07 45
TABLA 11: CASO DE USO CU-08 45
TABLA 12: CASO DE USO CU-09 46
TABLA 13: CASO DE USO CU-10 47
TABLA 14: CASO DE USO CU-11 47
TABLA 15: CASO DE USO CU-12 48
TABLA 16: CASO DE USO CU-13 48
TABLA 17: CASO DE USO CU-14 49
TABLA 18: PROTOTIPO DE TABLA PARA LA DEFINICIÓN DE REQUISITOS 49
TABLA 19: REQUISITO FUNCIONAL RF-01 50
TABLA 20: REQUISITO FUNCIONAL RF-02 51
TABLA 21: REQUISITO FUNCIONAL RF-03 51
TABLA 22: REQUISITO FUNCIONAL RF-04 51
TABLA 23: REQUISITO FUNCIONAL RF-05 52
TABLA 24: REQUISITO FUNCIONAL RF-06 52
TABLA 25: REQUISITO FUNCIONAL RF-07 53
TABLA 26: REQUISITO FUNCIONAL RF-08 53
TABLA 27: REQUISITO FUNCIONAL RF-09 53
TABLA 28: REQUISITO FUNCIONAL RF-10 54
TABLA 29: REQUISITO FUNCIONAL RF-11 54
TABLA 30: REQUISITO FUNCIONAL RF-12 54
TABLA 31: REQUISITO FUNCIONAL RF-13 55
TABLA 32: REQUISITO NO FUNCIONAL RNF-01 55
TABLA 33: REQUISITO NO FUNCIONAL RNF-02 55
TABLA 34: REQUISITO NO FUNCIONAL RNF-03 56
TABLA 35: REQUISITO NO FUNCIONAL RNF-04 56
TABLA 36: REQUISITO NO FUNCIONAL RNF-05 56
TABLA 37: REQUISITO NO FUNCIONAL RNF-06 57
TABLA 38: REQUISITO NO FUNCIONAL RNF-07 57
TABLA 39: REQUISITO DE RESTRICCIÓN RR-01 57
TABLA 40: REQUISITO DE RESTRICCIÓN RR-02 58
TABLA 41: REQUISITO DE RESTRICCIÓN RR-03 58
TABLA 42: REQUISITO DE RESTRICCIÓN RR-04 58
TABLA 43: REQUISITO DE RESTRICCIÓN RR-05 58
TABLA 44: MATRIZ DE TRAZABILIDAD CASOS DE USO - REQUISITOS 59
TABLA 45: COMPONENTE GUI 63
TABLA 46: COMPONENTE ACCESO SERVICIOS WEB 63
TABLA 47: COMPONENTE ENTIDADES 63
TABLA 48: COMPONENTE BARCODE SCANNER 64
TABLA 49: COMPONENTE GESTOR DE DATOS 64
TABLA 50: COMPONENTE SERVICIOS WEB 64
TABLA 51: COMPONENTE ACCESO A DATOS 65
TABLA 52: PRUEBA DE ACEPTACIÓN PA-01 89

TABLA 53: PRUEBA DE ACEPTACIÓN PA-02 90
TABLA 54: PRUEBA DE ACEPTACIÓN PA-03 91
TABLA 55: PRUEBA DE ACEPTACIÓN PA-04 92
TABLA 56: PRUEBA DE ACEPTACIÓN PA-05 93
TABLA 57: PRUEBA DE ACEPTACIÓN PA-06 94
TABLA 58: PRUEBA DE ACEPTACIÓN PA-07 95
TABLA 59: PRUEBA DE ACEPTACIÓN PA-08 96
TABLA 60: PRUEBA DE ACEPTACIÓN PA-09 96
TABLA 61: PRUEBA DE ACEPTACIÓN PA-10 97
TABLA 62: PRUEBA DE ACEPTACIÓN PA-11 98
TABLA 63: MATRIZ DE TRAZABILIDAD REQUISITOS-PRUEBAS 98
TABLA 64: PLANIFICACIÓN DE LAS TAREAS DEL PROYECTO 103
TABLA 65: COSTES DE PERSONAL 106
TABLA 66: COSTES DE HARDWARE 106
TABLA 67: COSTES DE SOFTWARE 107
TABLA 68: COSTE TOTAL DEL PROYECTO 107

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Distribución de versiones de Android a octubre de 2011 17
Ilustración 2: Arquitectura de Android 19
Ilustración 3: Ciclo de vida de una Actividad en Android 25
Ilustración 4: Construcción de un object JSON 29
Ilustración 5: Construcción de un array JSON 29
Ilustración 6: Formación de un string en JSON 30
Ilustración 7: Opciones válidas para la formación de un value en JSON 31
Ilustración 8: Formación de un number en JSON 31
Ilustración 9: Descripción esquemática del sistema 39
Ilustración 10: Diagrama de casos de uso 40
Ilustración 11: Modelo-vista-controlador en el sistema 60
Ilustración 12: Diagrama de despliegue del sistema 61
Ilustración 13: Diagrama de componentes del sistema 62
Ilustración 14: Diagrama de clases de la aplicación cliente 66
Ilustración 15: Diagrama de clases de la aplicación cliente dividido por zonas 67
Ilustración 16: Subdiagrama de clases 1 de la aplicación cliente 68
Ilustración 17: Subdiagrama de clases 2 de la aplicación cliente 69
Ilustración 18: Subdiagrama de clases 3 de la aplicación cliente 70
Ilustración 19: Subdiagrama de clases 4 de la aplicación cliente 71
Ilustración 20: Subdiagrama de clases 5 de la aplicación cliente 72
Ilustración 21: Subdiagrama de clases 6 de la aplicación cliente 73
Ilustración 22: Subdiagrama de clases 7 de la aplicación cliente 74
Ilustración 23: Diagrama de clases de la aplicación servidor 75
Ilustración 24: Diagrama de clases de la aplicación servidor dividido por zonas 76
Ilustración 25: Subdiagrama de clases 1 de la aplicación servidor 77
Ilustración 26: Subdiagrama de clases 2 de la aplicación servidor 78
Ilustración 27: Subdiagrama de clases 3 de la aplicación servidor 79
Ilustración 28: Subdiagrama de clases 4 de la aplicación servidor 80
Ilustración 29: Subdiagrama de clases 5 de la aplicación servidor 81
Ilustración 30: Diagrama de BBDD del sistema 82
Ilustración 31: Establecimiento de una conexión SSL segura 101
Ilustración 32: Diagrama de Gantt de la planificación 104
Ilustración 33: Pantalla de acceso de la aplicación 111
Ilustración 34: Pantalla de menú de usuario 112
Ilustración 35: Pantallas de inserción de nuevo elemento 112
Ilustración 36: Pantallas de consulta de inventario y lista de resultados 113
Ilustración 37: Pantallas de vista de elemento en detalle y de editar ubicación 113
Ilustración 38: Pantallas de elemento en detalle activo y borrado 114
Ilustración 39: Pantalla de consulta de operaciones del perfil usuario 114
Ilustración 40: Pantallas de lista de operaciones y vista en detalle 115
Ilustración 41: Pantalla de menú de usuario con botón de desconectar y pantalla principal 116
Ilustración 42: Pantallas de menú principal de administrador y consulta de operaciones 116
Ilustración 43: Pantallas de gestión de usuarios y lista de usuarios existentes 117
Ilustración 44: Pantallas de vista de usuario en detalle activo e inactivo 117
Ilustración 45: Pantallas de inserción de nuevo usuario y lista de usuarios existentes 118
Ilustración 46: Pantalla de gestión de recursos y lista de tipos de elemento 118
Ilustración 47: Pantallas de añadir nuevo tipo de elemento y lista actualizada 119
Ilustración 48: Pantalla de añadir ubicación 119
Ilustración 49: Pantallas de menú de administrador con botón de desconectar y pantalla principal 120

1. INTRODUCCIÓN

El objetivo de este capítulo es presentar el trabajo realizado en este Proyecto Fin de Carrera y describir la estructura del presente documento. En esta introducción, se describirá brevemente el problema, los motivos que han llevado a su desarrollo y los objetivos que se desean alcanzar.

1.1 MOTIVACIÓN

La motivación para la realización del proyecto desarrollado, y del que se hablará a lo largo de este documento, es la mejora sustancial de los procesos de gestión de inventario en pymes, grandes empresas e instituciones públicas (como puede ser una universidad). Dicha mejora es necesaria en los tiempos actuales, en los que es necesario que todos los procesos sean lo más ágiles y eficientes posible.

Otro factor que ha motivado la realización de este proyecto es la aceptación y expansión que tienen hoy en día los dispositivos móviles inteligentes (*smartphones*), y más aún, en el ámbito empresarial. La gran capacidad de procesamiento, almacenamiento y personalización con la que cuentan estos dispositivos, hace posible que hoy en día cualquier trabajador de cualquier corporación pueda llevar casi la totalidad de la información necesaria para la realización de su trabajo a donde quiera que vaya, sin tener que estar físicamente en su puesto de trabajo. Esto deriva a una gran flexibilidad y agilidad en los procesos laborales, ahorrando costes y gran cantidad de tiempo. Así pues, se desean aprovechar todas estas características para su aplicación en el proceso de gestión de inventario, clave en cualquier empresa.

Finalmente, otro aspecto fundamental para que este proyecto se lleve a cabo de manera viable, es el gran avance en infraestructura de red y descentralización de la información fundamental con la que cuentan las empresas hoy en día. Esto hace posible acceder de manera fiable y segura a la información de inventario de la empresa desde cualquier lugar donde sea necesario, ahorrando tiempo y recursos en procesos de actualización de datos posteriores a las actuaciones de inventariado.

En resumen, todos los factores expuestos hacen perfectamente viable, e incluso necesaria, la realización de este proyecto: existe la necesidad, se cuenta con los recursos necesarios para su desarrollo y no se detectan factores importantes que limiten su aplicación.

1.2 OBJETIVOS

El objetivo principal que persigue este Proyecto Fin de Carrera es la gestión del proceso de inventario de una compañía, de manera eficiente, segura, sencilla e intuitiva para los usuarios.

Para alcanzar este objetivo principal, existen otros objetivos secundarios a conseguir:

- Análisis previo de los procesos principales en la gestión de inventario, que serán implementados en el sistema.

- Análisis de la información necesaria que se almacenará en una base de datos, que actuará como base a una base de datos corporativa real.

- Diseño e implementación de la base de datos que contemple la información obtenida en el punto anterior.

- Diseño y construcción de una aplicación que ejecute en un ordenador que actuará a modo de servidor del sistema, y que sea capaz de acceder de manera correcta a la información contenida en la BBDD, y además comunicarse de manera efectiva con los dispositivos móviles, recibiendo peticiones y proporcionando la información requerida.

- Conocer los detalles de la arquitectura de Android, así como obtener los conocimientos necesarios para desarrollar aplicaciones para ella[1].

- Diseñar e implementar una aplicación Android de gestión de inventario, que implemente los procesos principales identificados en el primer punto, que se comunique con el servidor de manera eficaz y eficiente, y presente la información al usuario de manera sencilla e intuitiva.

- Realizar pruebas de manera exhaustiva a la finalización de la implementación del sistema, para asegurar de manera fehaciente que se ha tenido éxito en la construcción del sistema, cumpliendo los objetivos marcados y realizando los procesos necesarios de manera correcta.

- Documentar el código y todo el proceso de realización del proyecto, para facilitar la comprensión del mismo a las personas no involucradas en el mismo, y que pudieran tener interés en su uso para aplicarlo a los procesos de inventario de su corporación.

1.3 ESTRUCTURA DEL DOCUMENTO

El presente documento está dividido en varios capítulos, agrupados de forma lógica de tal modo que a través de su lectura ordenada se consiga una comprensión paulatina y global de la aplicación y su dominio:

En el **capítulo 1, Introducción**, se presenta la motivación del proyecto, los objetivos que se pretenden alcanzar, y la estructura del presente documento. En resumen, se pretende dar una visión inicial y global del proyecto.

En el **capítulo 2, Estado del arte**, se presentan y analizan a fondo las tecnologías que se utilizarán en la realización del proyecto, así como una justificación de su uso frente a otras alternativas disponibles.

En el **capítulo 3, Análisis del sistema**, se estudia detenidamente la funcionalidad requerida por la aplicación, dando lugar a los casos de uso y a los requisitos del sistema, herramienta fundamental para tener presente hacia dónde se pretende llegar con el proyecto, y que servirán también a la finalización del mismo, para verificar que el proyecto se ha llevado a cabo de manera exitosa.

En el **capítulo 4, Diseño e implementación**, se define con precisión la aplicación a todos los niveles, desde su arquitectura general, hasta llegar al nivel de detalle de clase, exponiendo de manera precisa la disposición de todos los elementos del sistema y la comunicación entre los mismos. Se presentan también, tanto el diseño detallado de la base de datos utilizada, como algunos aspectos interesantes del proceso de implementación y consideraciones a tener en cuenta.

En el **capítulo 5, Pruebas**, se muestra detalladamente la batería de pruebas realizada al sistema en su totalidad y que permitirá validar los requisitos identificados en la fase de análisis, y concluir el proyecto con éxito.

En el **capítulo 6, Conclusiones y líneas futuras**, se recogerán las conclusiones posteriores a la realización del proyecto, así como una lista de posibles aspectos ampliables en versiones posteriores, o líneas de investigación en las que se puede trabajar para la mejora constante de la aplicación.

En el **capítulo 7, Planificación y presupuesto**, se expondrá la planificación detallada que se seguirá en el desarrollo del proyecto, así como el presupuesto, que se explicará por partes, justificando los costes de personal, hardware y software, e informando del coste total del proyecto, incluyendo beneficios y margen de riesgo.

En el **capítulo 8, Bibliografía**, se mostrarán todas las fuentes de información consultadas durante el proceso de desarrollo del proyecto.

Finalmente, en el **Anexo I, Manual de usuario**, se enseñará a utilizar la aplicación, mostrando gráficamente todas las opciones disponibles, y explicando cómo hay que actuar con la aplicación para realizar los procesos de inventario de manera satisfactoria.

1.4 GLOSARIO DE TÉRMINOS

Actividad	Cada una de las pantallas que puede ver el usuario dentro de una aplicación Android.
API	Interfaz de Programación de Aplicaciones, es un conjunto de funciones que ofrece una librería para ser utilizado por otro software.
Hash	Construcción criptográfica empleada en aplicaciones para evitar la recuperación de información sensible.
IDE	Un entorno de desarrollo integrado (en inglés *Integrated Development Environment*) es un programa informático compuesto por un conjunto de herramientas de programación.
IVA	Impuesto al valor añadido.
JSON	*JavaScript Object Notation*, formato ligero para el intercambio de datos.
MVC	Modelo Vista Controlador, patrón de diseño de Ingeniería del Software que separa los datos de una aplicación, la interfaz y la lógica de control en tres componentes diferentes.
REST	Es una técnica de arquitectura software para sistemas hipermedia distribuidos como la *World Wide Web*.
SDK	*Software Development Kit* o Kit de Desarrollo de Software, se trata de un conjunto de herramientas de desarrollo que permiten a un programador crear aplicaciones para un sistema concreto.
SGBD	Sistema de Gestión de Bases de Datos, es un tipo de software muy específico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Tabla 1: Glosario de términos

2. ESTADO DEL ARTE

2.1 ANDROID

Android es una plataforma de *software* y un sistema operativo para dispositivos móviles basada en un *kernel Linux*, desarrollada por *Google* y más tarde por la *Open Handset Alliance*[2]. Esta plataforma permite a los desarrolladores escribir código en *Java* que se ejecuten en móviles mediante las librerías *Java* desarrolladas por *Google*. También se pueden escribir aplicaciones en otros lenguajes, como por ejemplo *C*, para posteriormente ser compiladas en código nativo ARM y ejecutarlas, aunque este proceso de desarrollo no está soportado oficialmente por *Google*. La mayor parte de la plataforma de *Android* está disponible bajo licencia de software libre de *Apache* y otras licencias de código abierto.

La estructura del sistema operativo, compuesto por más de 12 millones de líneas de código escritas en C, C++, Java y XML, está formada por un conjunto de aplicaciones que se ejecutan en un *framework* Java[3] de aplicaciones orientadas a objetos sobre una máquina virtual conocida como Dalvik Virtual Machine[4] y que se compilan en tiempo de ejecución.

Breve Historia

En Julio de 2005, *Google* adquirió *Android, Inc*, una pequeña *startup* de California. En esos momentos, la compañía se dedicaba a la creación de *software* para teléfonos móviles. Una vez en Google, el equipo desarrolló un S.O. basado en *Linux* para dispositivos móviles. Más adelante, Google adaptó su buscador y sus aplicaciones para su uso en móviles.

En septiembre del 2007, *Google* tenía varias patentes de aplicaciones sobre el área de la telefonía móvil. El 5 de noviembre del mismo año, se anunció la fundación de la *Open Handset Alliance* al mismo tiempo que la creación de la plataforma *Android.* La *Open Handset Alliance* está formada por un consorcio de 34 compañías de hardware, software y telecomunicaciones, entre las cuales se incluyen *Google, HTC, Intel* y *Motorola* entre otras, dedicadas a investigar estándares abiertos para dispositivos móviles.

El primer teléfono en el mercado que posee *Android* es el *T-Mobile G1* (también conocido como *Dream),* lanzado el día 22 de octubre de 2008 que viene con la versión *Android* 1.0 preinstalada. Este móvil es el resultado conjunto de *T-Mobile*, *HTC* y *Google.* Por último, desde el 21 de octubre de 2008, *Android* está disponible como código abierto. Gracias a esto, cualquiera puede añadir extensiones, nuevas aplicaciones o reemplazar las existentes por otras dentro del dispositivo móvil.

Versiones

La primera versión estable, la 1.0, se lanzó el 23 de septiembre de 2008. Desde entonces han salido otras 7 versiones, que se identifican por un número y un nombre de postre que cuya inicial sigue el orden alfabético, como se puede observar en la siguiente tabla:

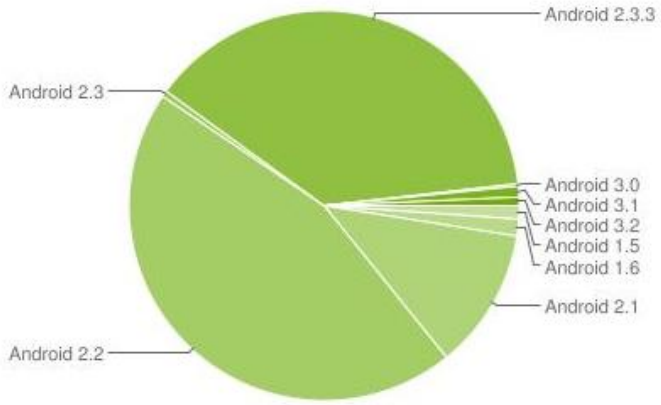
Número de versión	Nombre de la versión	Fecha de salida
1.0	-	23 de septiembre de 2008
1.1	-	9 de febrero de 2009
1.5	**C**upcake	30 de abril de 2009
1.6	**D**onut	15 de septiembre de 2009
2.0/2.1	**É**clair	26 de octubre de 2009
2.2	**F**royo	20 de mayo de 2010
2.3	**G**ingerbread	6 de diciembre de 2010
3.0/3.1	**H**oneycomb	27 de enero de 2011
4.0	**I**ce Cream Sandwich	19 de octubre de 2011

Tabla 2: Versiones del sistema operativo Android

A pesar de que en un principio Android estaba orientado a los teléfonos móviles, la última versión, *Honeycomb*, está diseñada para funcionar en *tablets* y *netbooks*. Se espera que Google unifique en un futuro el soporte de todos los dispositivos en un único sistema.

Una de las ventajas de Android, que a su vez puede suponer un problema para los desarrolladores, es la diversidad de los terminales que utilizan su sistema operativo. Al contar con soporte para hardware tan diferente, la compatibilidad de las aplicaciones en algunos casos se ve reducida, y es necesario diseñar interfaces de diferentes tamaños, contar con que algunas funciones como la cámara, el GPS o los acelerómetros no siempre van a estar disponibles en los terminarles, etc. Google ha facilitado herramientas para resolver alguno de estos problemas, pero otros simplemente no tienen solución. También se ha de tener en cuenta que no todos los móviles tienen la misma versión del sistema operativo instalada, y que por tanto, algunas de las funciones del

API de Android no son compatibles. A continuación se puede apreciar el porcentaje de dispositivos que cuentan con cada versión del sistema:



Plataforma	Nivel de API	%
Android 1.5	3	1,1%
Android 1.6	4	1,4%
Android 2.1	7	11,7%
Android 2.2	8	45,3%
Android 2.3 – 2.3.2	9	0,5%
Android 2.3.3–2.3.4	10	38,2%
Android 3.0	11	0,2%
Android 3.1	12	0,9%
Android 3.2	13	0,7%

Ilustración 1: Distribución de versiones de Android a octubre de 2011

La acogida de este sistema operativo móvil por parte de los fabricantes ha sido enorme, actualmente existen cientos de modelos con Android. Son varios los motivos de este éxito. Por un lado el sistema es libre, y las compañías pueden usarlo en sus teléfonos sin pagar licencias a Google y cuentan con el derecho a realizar todas las modificaciones que quieran, lo que les permite personalizarlo a su antojo. Por otro lado es un sistema atractivo para los usuarios, ya que cuenta con cientos de miles de aplicaciones disponibles, en su mayoría gratuitas.

Las aplicaciones para estos dispositivos se pueden obtener del *Android Market*, la tienda de aplicaciones oficial de Google para Android. Esta tienda es una de las ventajas del desarrollador, le permite una vez pagada la licencia de desarrollo (cuesta 25 dólares americanos), publicar su trabajo y que de esta forma sea descargable desde cualquier móvil. De esta forma, el conjunto de los usuarios potenciales de una aplicación es inmenso, y como se estimó en junio de 2011 hay 500.000 nuevos cada día.

Respecto a las herramientas de desarrollo, Google ofrece una web desde la que se puede descargar cualquiera de las versiones del SDK, y en la que se detalla como configurar el conocido entorno de desarrollo de código libre *Eclipse* y las Herramientas de Desarrollo Android (ADT)[5].

Características de Android

- Amplia variedad de diseños (*VGA*, librerías de gráficos 2D y 3D…).
- Almacenamiento de datos en BBDD *SQLite*[6].
- Conectividad (*GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth y Wi-F*i).
- Mensajería (*SMS y MMS*).
- Navegador Web.
- Máquina virtual de *Java*.
- Las aplicaciones escritas en *Java* pueden ser compiladas y ejecutadas en la maquina virtual de *Dalvik*, la cual es una especializada máquina virtual diseñada para uso en dispositivos móviles.
- Soporte de formatos (*MPEG-4, H.264, MP3, AAC, OGG, AMR, JPEG, PNG, GIF*).
- Soporte para hardware adicional (cámaras de video, pantallas táctiles, GPS, acelerómetros…).
- Entorno de desarrollo (emulador, herramientas de depuración, perfiles de memoria y funcionamiento, *plugin* para Eclipse IDE[7]).

Arquitectura de Android

En la siguiente figura se muestra la composición de la arquitectura de Android.

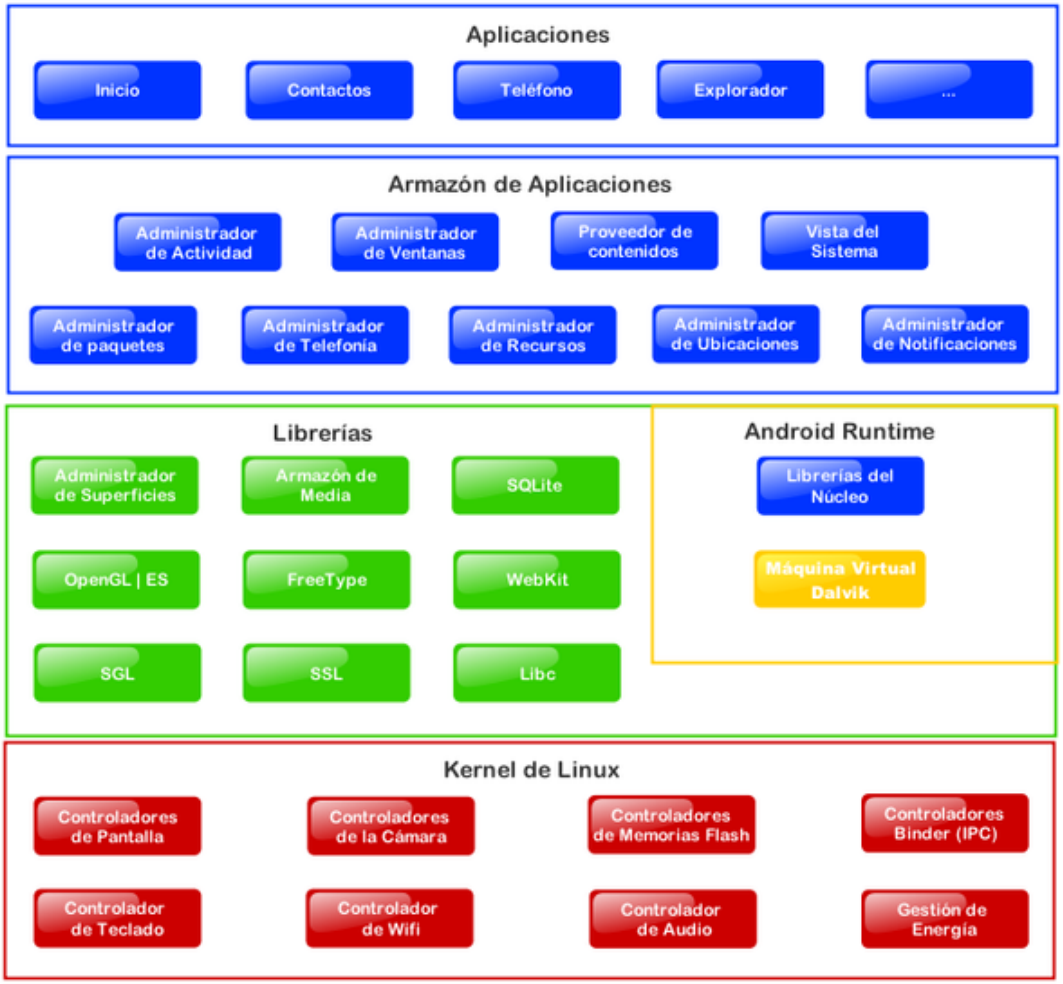


Ilustración 2: Arquitectura de Android

A continuación, se detallarán los diferentes componentes principales de la arquitectura de *Android*:

- Aplicaciones: Las aplicaciones base incluirán un cliente de email, programa de SMS, calendario, mapas, navegador, contactos, y otros. Todas las aplicaciones están escritas en el lenguaje de programación *Java*.

- *Framework* de aplicaciones: Los desarrolladores tienen acceso completo a las *API*s del *framework* usado por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del *framework*). Éste mismo mecanismo permite que los componentes sean reemplazados por el usuario. Este *framework* está formado por:

  - Un extenso conjunto de *Vistas* tales como listas, cajas de texto, botones…
  - *Content Providers* que permiten a las aplicaciones acceder a información de otras aplicaciones o compartir su propia información.
  - *Resource Manager*, que proporciona acceso a recursos que no son código, como pueden ser gráficos, cadenas de texto...
  - *Notification Manager*, que permite a las aplicaciones mostrar alarmas personalizadas en la barra de estado.
  - *Activity Manager*, que gestiona el ciclo de vida de las aplicaciones.

- Librerías: *Android* incluye un set de librerías C/C++ usadas por varios componentes del sistema. Estas capacidades se exponen a los desarrolladores a través del *framework* de aplicaciones de *Android*, el cual interactúa con las librerías mediante *JNI* (*Java Native Interface).* Algunas son: *System C library* (implementación librería *C* estándar), librerías de medios, librerías de gráficos, 3d, *SQLite*, entre otras.

- *Runtime* de *Android*: *Android* incluye un set de librerías base que proveen la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación *Java*. Cada aplicación *Android* corre su propio proceso, con su propia instancia de la máquina virtual *Dalvik*. *Dalvik* ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. *Dalvik* ejecuta archivos en el formato *Dalvik Executable* (*.dex*), el cual está optimizado para memoria mínima.

- Núcleo - *Linux*: *Android* depende de un *Linux* versión 2.6 para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red, y modelo de *drivers*. El núcleo también actúa como una capa de abstracción entre el *hardware* y el resto de la pila.

Anatomía de una aplicación de Android

Dentro de una aplicación de *Android* hay cuatro componentes principales: *Activities, Listeners, Services y Content Providers*. Todas las aplicaciones de *Android* están formadas por algunos de estos elementos o combinaciones de ellos.

Activity

Las *Activities* (o Actividades) son el elemento constituyente de *Android* más común. Para implementarlas se utiliza una clase por cada Actividad que extiende de la clase base *Activity*[8]. Cada clase mostrará una interfaz de usuario, compuesta por *Views* (o Vistas). Cada vez que se cambie de Vista, se cambiará de Actividad, como por ejemplo en una aplicación de mensajería que se tiene una Vista que muestra la lista de contactos y otra Vista para escribir los mensajes. Cuando cambiamos de Vista, la anterior queda pausada y puesta dentro de una pila de historial para poder retornar en caso necesario. También se pueden eliminar las Vistas del historial en caso de que no se necesiten más. Para pasar de vista en vista, *Android* utiliza una clase especial llamada *Intent*.

Un *Intent*[9] es un objeto mensaje y que, en general, describe qué quiere hacer una aplicación. Las dos partes más importantes de un *Intent* son la acción que se quiere realizar y la información necesaria que se proporciona para poder realizarla, la cual se expresa en formato *URI*. Un ejemplo sería ver la información de contacto de una persona, la cual mediante un *Intent* con la acción *ver* y la URI que representa a esa persona se podría obtener. Relacionado con los *Intents*, hay una clase llamada *IntentFilter* que es una descripción de qué *Intents* puede una gestionar un *Activity*. Mediante los *IntentFilters*, el sistema puede resolver *Intents*, buscando cuáles posee cada actividad y escogiendo aquel que mejor se ajuste a sus necesidades. El proceso de resolver *Intents* se realiza en tiempo real, lo cual ofrece dos beneficios:

- Las actividades pueden reutilizar funcionalidades de otros componentes simplemente haciendo peticiones mediante un *Intent.*

- Las actividades pueden ser remplazadas por nuevas actividades con *IntentFilters* equivalentes.

Listeners

Los *Listeneres* se utilizan para reaccionar a eventos externos (por ejemplo, una llamada). Los *Listeners* no tienen *UI* (*User Interface*), pero pueden utilizar el servicio *NotificationManager* para avisar al usuario. Para lanzar un aviso no hace falta que la aplicación se esté ejecutando, en caso necesario, *Android* la iniciará si se activa el *Listeners* por algún evento.

Services

Un *Service*, o Servicio, es básicamente un código que se ejecuta durante largo tiempo y sin necesidad de *UI*, como puede ser un gestor de descarga, en el cual se indican los contenidos a descargar y posteriormente el usuario puede acceder a una

nueva vista sin que el gestor se interrumpa. En caso de que haya múltiples servicios a la vez, se les puede indicar diferentes prioridades según las necesidades.

Content Providers

En *Android*, las aplicaciones pueden guardar su información en ficheros, BBDD *SQLite*, etc., pero en caso de que lo que se quiera sea compartir dicha información con otras aplicaciones, lo necesario es un *Content Provider*. Un *Content Provider* es una clase que implementa un conjunto estándar de métodos que permite a otras aplicaciones guardar y obtener la información que maneja dicho *Content Provider*.

Android Manifest

En *Android* existe un archivo *XML* llamado *AndroidManifest* que, aunque no forme parte del código principal de la aplicación, es necesario para su correcto funcionamiento. Este archivo es el fichero de control que le dice al sistema qué tiene que hacer con todos los componentes anteriormente mencionados en este apartado que pertenecen a una aplicación en concreto.

Tipos de aplicación de Android

Un *Android Package* (.apk) es el fichero que contiene el código de la aplicación y sus recursos, y que posteriormente se instala en el dispositivo para poder ejecutar la aplicación.

Tareas

Una tarea en *Android* es lo que el usuario ve como una aplicación y el desarrollador ve como una o más Actividades donde el usuario interacciona y va pasando de Vista en Vista. Dentro de las tareas, una actividad toma el papel de punto de entrada (será la primera en mostrarse cuando se ejecute la aplicación) y las demás, si hay, formarán parte de la misma tarea, a la espera de ser instanciadas.

Procesos

En *Android*, los procesos se ejecutan a nivel de *kernel* y el usuario normalmente no tiene constancia de ellos. Todo el código de la aplicación se suele ejecutar en un proceso dedicado pero también se puede especificar si sólo se quiere que se ejecute en el proceso una determinada clase o componente de la aplicación. Los principales usos de los procesos son:

- Mejorar la estabilidad o seguridad de las aplicaciones.

- Reducir la sobrecarga de proceso ejecutando el código de múltiples aplicaciones en el mismo proceso.

- Ayudar al sistema a gestionar los recursos separando partes de código pesado en un proceso separado que puede ser eliminado independientemente de otras partes de la aplicación.

Threads

En lo referente a *threads*, *Android* evita la creación de *threads* adicionales por parte de un proceso, manteniendo la aplicación en un sólo *thread*. Esto repercute de manera importante en las llamadas a instancias de Actividades, *Listeners* y Servicios, ya que sólo pueden ser hechas por el *thread* principal del proceso en el que están corriendo. Por otro lado, al no crearse un *thread* por cada instancia, dichas instancias no deben realizar operaciones largas o bloqueantes cuando son llamadas, de lo contrario, bloquearían todos los demás componentes del proceso.

Ciclo de vida de una aplicación de Android

Cada aplicación de *Android* corre en su propio proceso, el cual es creado por la aplicación cuando se ejecuta, y permanece hasta que la aplicación deja de trabajar o el sistema necesita memoria para otras aplicaciones. Una característica fundamental de *Android* es que el ciclo de vida de una aplicación no está controlado por la misma aplicación sino que lo determina el sistema a partir de una combinación de estados como pueden ser qué aplicaciones están funcionando, qué prioridad tienen para el usuario y cuánta memoria queda disponible en el sistema. De esta manera, *Android* sitúa cada proceso en una jerarquía de "importancia" basada en los estados comentados, como se puede ver a continuación.

- Un proceso en primer plano es uno que se requiere para lo que el usuario está actualmente haciendo. Se considera en primer plano si:

    - Esta ejecutándose una Actividad perteneciente a la pantalla con la que el usuario está interactuando.

    - Está ejecutando un *BroadcastReceiver*.

    - Está ejecutándose un servicio.

- Un proceso visible es aquel que contiene una Actividad que es visible al usuario mediante la pantalla, pero no en primer plano (está pausada). Este proceso solo se eliminará en caso de que sea necesario para mantener ejecutándose los procesos en primer plano.

- Un proceso de servicio es aquel que contiene un servicio que ha sido inicializado. No son directamente visibles al usuario y el sistema los mantendrá a no ser que no pueda servir los dos anteriores.

- Un proceso en *background* es aquel que acoge una actividad que no está actualmente visible al usuario. Mientras que dichos procesos implementen bien su propio ciclo de vida, el sistema puede eliminarlos para dar memoria a cualquiera de los 3 procesos anteriores.

- Un proceso vacío es aquel que no contiene ningún componente activo de ninguna aplicación. La única razón para mantener dicho proceso es para mejorar sus inicializaciones posteriores a modo de caché.

Para comprender mejor el ciclo de vida de una aplicación de *Android*, en la siguiente figura se muestra el diagrama de flujo de dicho ciclo, mencionando también los métodos que se llaman durante el transcurso del mismo.

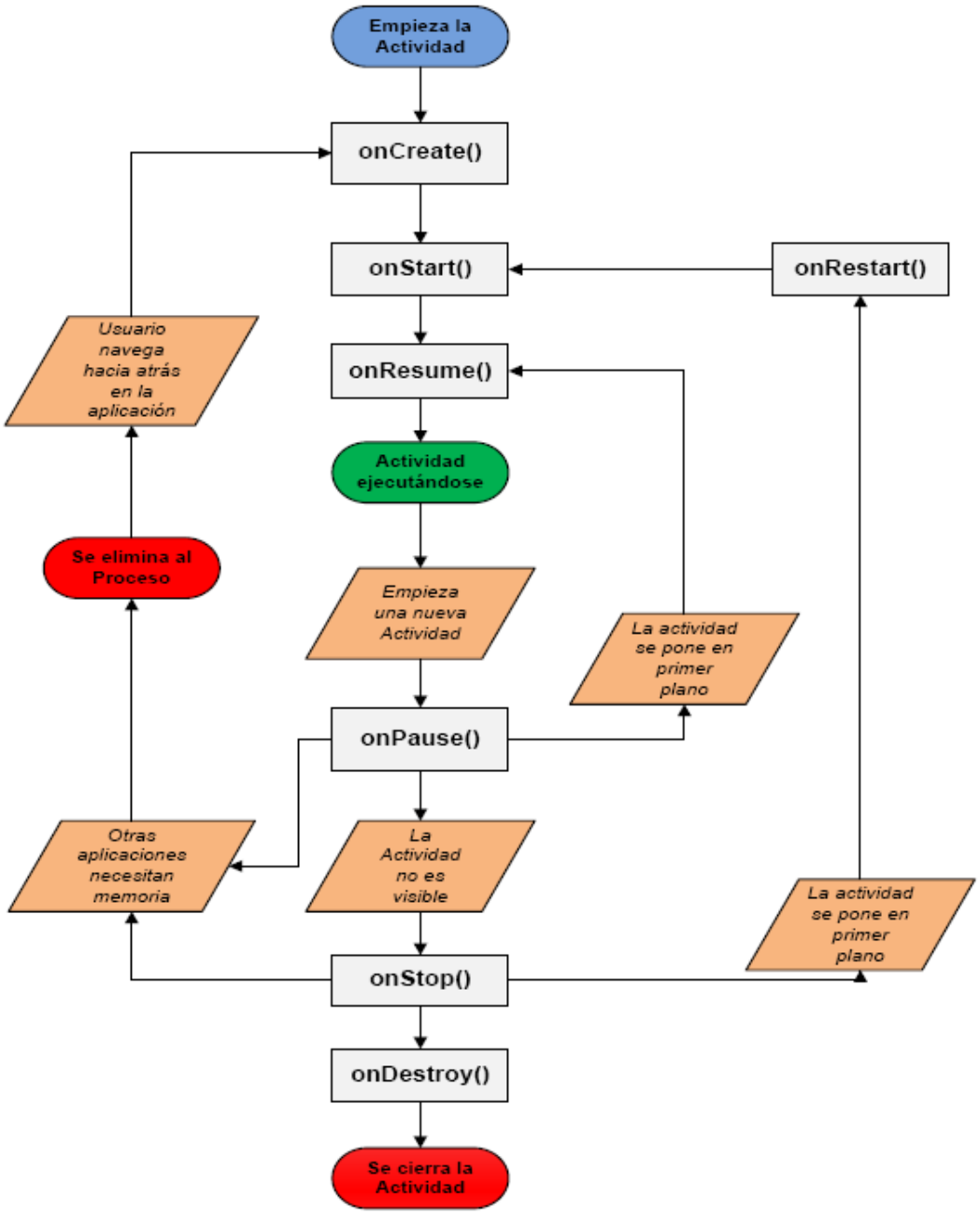


Ilustración 3: Ciclo de vida de una Actividad en Android

2.2 ECLIPSE

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado *Eclipse Modeling Project*, cubriendo casi todas las áreas de *Model Driven Engineering*.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para *VisualAge*. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la *Common Public License*, pero después fue re-licenciado bajo la *Eclipse Public License*. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con la Licencia pública general de GNU (GNU GPL).

Arquitectura

La base para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- Plataforma principal - inicio de Eclipse, ejecución de plugins.
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto.
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes.

Los widgets de Eclipse están implementados por una herramienta de widget para Java llamada SWT, a diferencia de la mayoría de las aplicaciones Java, que usan las

opciones estándar *Abstract Window Toolkit* (AWT) o *Swing*. La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada *JFace*, la cual simplifica la construcción de aplicaciones basadas en SWT.

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y sistemas de gestión de base de datos. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene por qué ser usado únicamente para soportar otros lenguajes de programación.

La definición que da el proyecto Eclipse acerca de su software es: "*una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular*".

En cuanto a las aplicaciones clientes, eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. Por ejemplo, GEF (*Graphic Editing Framework* - Framework para la edición gráfica) es un plugin de Eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto *wysiwyg* hasta editores de diagramas UML, interfaces gráficas para el usuario (GUI), etc. Dado que los editores realizados con GEF "viven" dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Mediante diversos plugins estas herramientas están también disponibles para otros lenguajes como C/C++ (Eclipse CDT) y en la medida de lo posible para lenguajes de script no tipados como *PHP* o *Javascript*. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadata en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

Características

Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con *JUnit*, control de versiones con *CVS*,

integración con *Ant*, asistentes (*wizards*) para creación de proyectos, clases, tests, etc., y refactorización.

Asimismo, a través de plugins libremente disponibles es posible añadir control de versiones con *Subversion* e integración con *Hibernate*.

Versiones

Resumen de las versiones de Eclipse:

Número de versión	Nombre de la versión	Fecha de salida
3.0	Eclipse 3.0	28 de junio de 2004
3.1	Eclipse 3.1	28 de junio de 2005
3.2	Calisto	30 de junio de 2006
3.3	Europa	29 de junio de 2007
3.4	Ganímedes	25 de junio de 2008
3.5	Galileo	24 de junio de 2009
3.6	Helios	23 de junio de 2010
3.7	Índigo	22 de junio de 2011
4.2	Juno	Junio de 2012 (prevista)

Tabla 3: Versiones de Eclipse

2.3 JSON

JSON[10] se corresponde con las siglas *JavaScript Object Notation* y es un formato ligero para el intercambio de datos entre distintas partes. Es un lenguaje muy sencillo de leer y escribir por las personas, y fácil de ser analizado sintácticamente y ser generado por máquinas. Aunque es un subconjunto de la notación de JavaScript, JSON

es un formato de texto completamente independiente de JavaScript o cualquier otro lenguaje de programación.

Los mensajes JSON se construyen siguiendo dos estructuras:

- Una colección de pares nombre/valor. En varios lenguajes, esto equivale a un *object*, *record*, *struct*, *dictionary*, *hash table*, *keyed list* o *associative array*.

- Una lista de valores ordenados. En muchos lenguajes, equivale a un *array*, *vector*, *list* o *sequence*.

En JSON, los mensajes toman la forma de *object* y *array* respectivamente, y se describen del siguiente modo:

Un *object* es un conjunto no ordenado de pares nombre/valor. Un *object* comienza con '{' (llave izquierda) y termina con '}' (llave derecha). Cada nombre es seguido por ':' (dos puntos) y el par nombre/valor separados por ',' (coma).

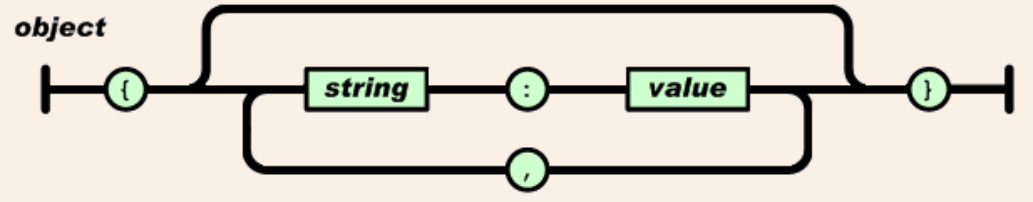


Ilustración 4: Construcción de un object JSON

Un *array* es una colección ordenada de valores. Un *array* empieza con el carácter [(corchete izquierdo) y termina con] (corchete derecho). Los valores están separados entre sí por , (coma).

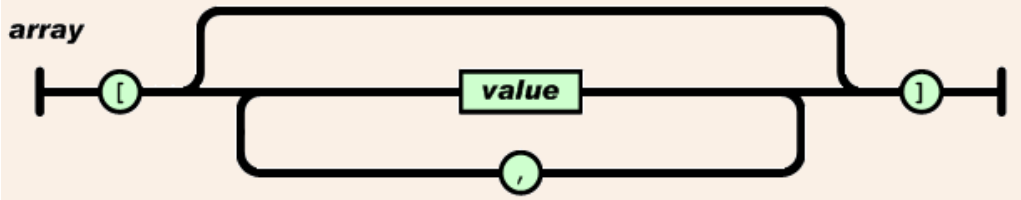


Ilustración 5: Construcción de un array JSON

Los *objects* y *arrays* necesitan de *strings* y *values* para contener información útil. Se forman del siguiente modo:

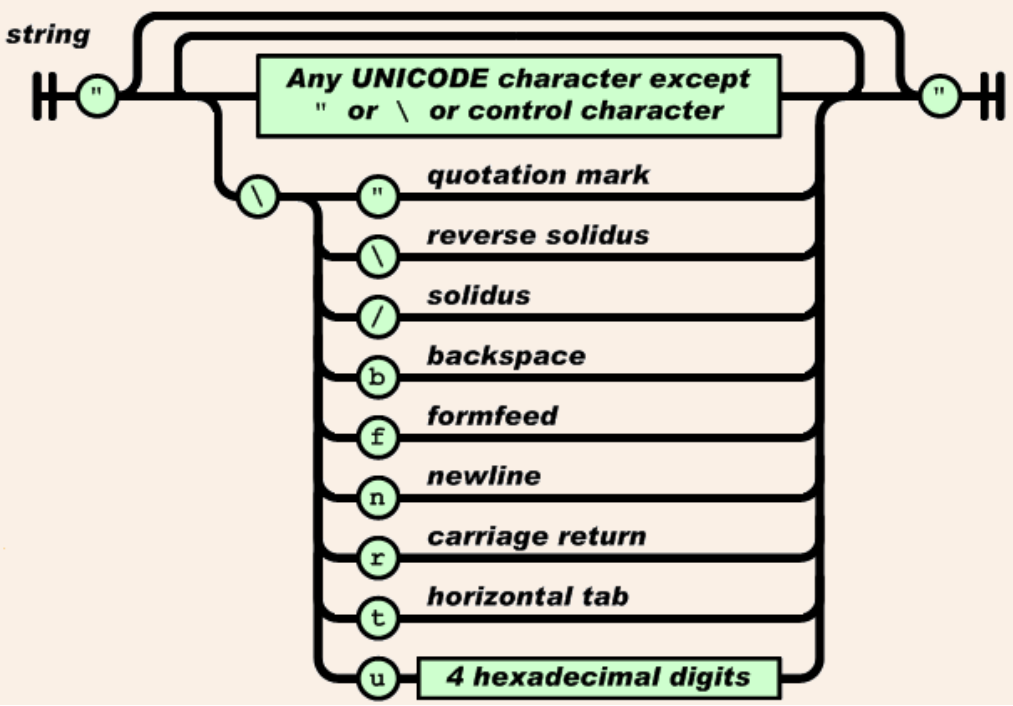


Ilustración 6: Formación de un string en JSON

Un *string* es una colección de cero o más caracteres Unicode, envueltos entre " (comillas dobles). Un *string* de JSON es muy similar a uno de C o Java, pudiéndose utilizar escapes de barra invertida (\).

Un *value* puede ser un *string* entre ‘"’ (comillas dobles), un *number*, *true* o *false* o *null*, un *object* o un *array*. Estas estructuras pueden anidarse.

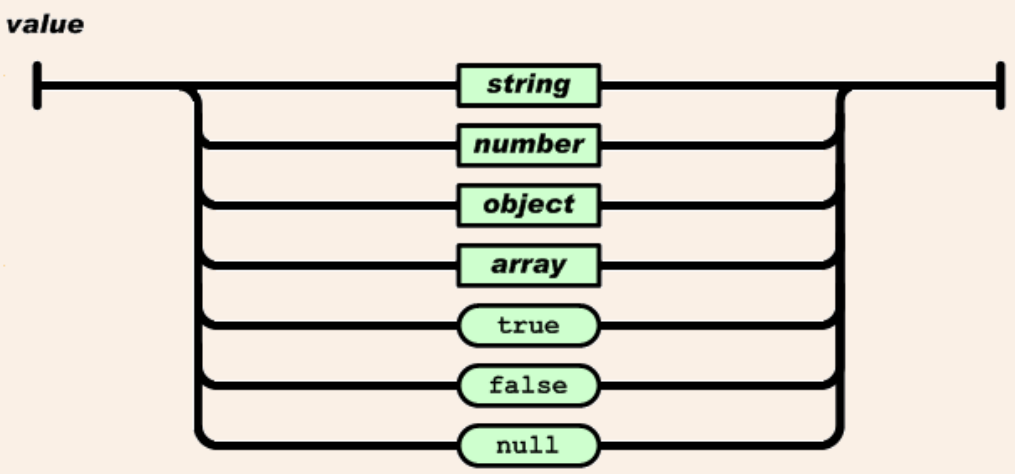


Ilustración 7: Opciones válidas para la formación de un value en JSON

Un *number* es muy similar a un número de Java o C, exceptuando que el formato octal y hexadecimal no se utilizan.

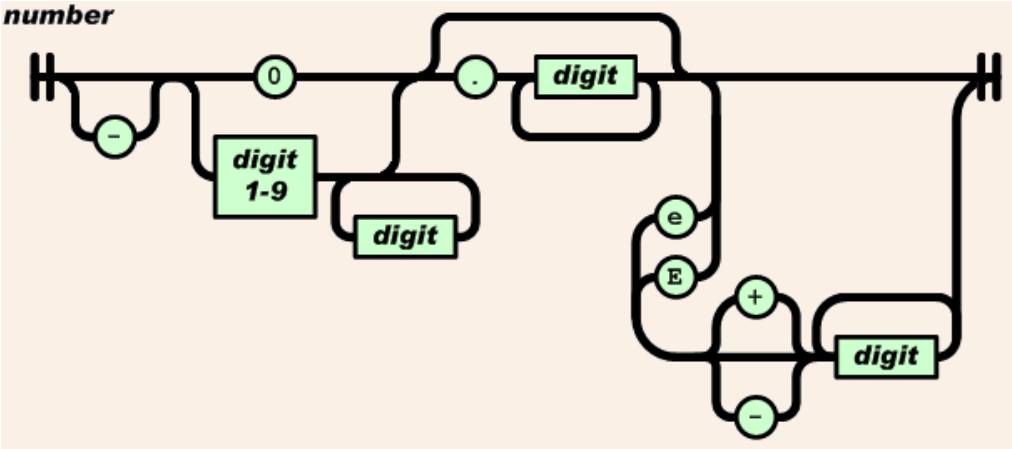


Ilustración 8: Formación de un number en JSON

2.4 REST WEB SERVICES

La Transferencia de Estado Representacional (Representational State Transfer) o *REST*[11] es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Si bien el término *REST* se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza *XML* y *HTTP*, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web *SOAP*. Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural REST de Fielding y también es posible diseñar interfaces *XMLHTTP* de acuerdo con el estilo de llamada a procedimiento remoto pero sin usar SOAP. Estos dos usos diferentes del término *REST* causan cierta confusión en las discusiones técnicas, aunque RPC no es un ejemplo de REST.

Los sistemas que siguen los principios REST se llaman con frecuencia *RESTful*; los defensores más acérrimos de REST se llaman a sí mismos *RESTafaris*.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un protocolo cliente/servidor sin estado: cada mensaje *HTTP* contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en *HTTP* utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de *URLs*, no son permitidas por REST).

- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: *HTTP* en sí define un conjunto pequeño de operaciones, las más importantes son *POST*, *GET*, *PUT* y *DELETE*. Con frecuencia estas operaciones se equiparan a las operaciones CRUD que se requieren para la persistencia de datos, aunque *POST* no encaja exactamente en este esquema.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su *URI*.

- El uso de hipermedias, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente *HTML* o *XML*. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

Recursos

Un concepto importante en REST es la existencia de *recursos* (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso o *URI*, de sus siglas en inglés). Para manipular estos recursos, los *componentes* de la red (clientes y servidores) se comunican a través de una interfaz estándar (*HTTP*) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían).

La petición puede ser tramitada por cualquier número de conectores (por ejemplo clientes, servidores, cachés, túneles, etc.) pero cada uno lo hace sin "ver más allá" de su propia petición (lo que se conoce como separación en capas, otra restricción de REST, que es un principio común con muchas otras partes de la arquitectura de redes y de la información). Así, una aplicación puede interactuar con un recurso conociendo el identificador del recurso y la acción requerida, no necesitando conocer si existen cachés, *proxys*, cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información. La aplicación, sin embargo, debe comprender el formato de la información devuelta (la representación), que es por lo general un documento HTML o XML, aunque también puede ser una imagen o cualquier otro contenido.

2.5 MYSQL

MySQL es un sistema de gestión relacional, multihilo y multiusuario con más de seis millones de instalaciones. El objetivo de la empresa en el momento de su creación en 1995 fue seguir con el estándar *SQL*, sin sacrificar la velocidad, fiabilidad o usabilidad. A lo largo de los años ha ido convirtiéndose en el gestor más popular del mundo por su rendimiento, eficiencia y facilidad de uso.

Sus características más importantes son:

- Implementación multihilo.
- Soporte en el tipo de datos.
- Portabilidad entre sistemas.
- Buen nivel de seguridad.

Su punto fuerte es una gran aceptación, debido a la cual existen infinidad de librerías para una gran cantidad de lenguajes de programación. En cuanto a la licencia, MySQL es de uso libre y código abierto pero Oracle tiene el copyright sobre ese código.

Hay varios tipos de gestión dentro del propio MySQL, entre los que destacan dos, *MyISAM* y *InnoDB*. En el caso de *MyISAM*, tecnología usada por defecto, se da una rápida lectura al utilizar el motor no transaccional ya que no tiene que hacer comprobaciones de integridad referencial ni bloquear las tablas por ausencia de atomicidad, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En segundo caso, *InnoDB*, es la tecnología opuesta, soporta las transacciones tipo *ACID* y bloqueo de registros junto con la integridad referencial, ofreciendo una fiabilidad y consistencia a cambio de un rendimiento menor.

2.6 HIBERNATE

Hibernate es una herramienta de Mapeo objeto-relacional (*ORM*) para la plataforma Java (y disponible también para .Net con el nombre de *NHibernate*) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (*XML*) o anotaciones en los *beans* de las entidades que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia *GNU LGPL*.

Características

Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr este objetivo, permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la *POO* (Programación Orientada a Objetos). Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por *SQL*. Hibernate genera las sentencias *SQL* y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

Hibernate ofrece también un lenguaje de consulta de datos llamado *HQL* (*Hibernate Query Language*), al mismo tiempo que una *API* para construir las consultas programáticamente (conocida como *"criteria"*).

Hibernate para Java puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente *Hibernate Annotations* que implementa el estándar JPA, que es parte de esta plataforma.

Historia

Hibernate fue una iniciativa de un grupo de desarrolladores dispersos alrededor del mundo conducidos por Gavin King.

Tiempo después, JBoss Inc. (empresa comprada por Red Hat) contrató a los principales desarrolladores de Hibernate y trabajó con ellos en brindar soporte al proyecto.

La rama actual de desarrollo de Hibernate es la 3.x, la cual incorpora nuevas características, como una nueva arquitectura *Interceptor/Callback*, filtros definidos por el usuario, y (opcionalmente) el uso de anotaciones para definir la correspondencia en

lugar de (o conjuntamente con) los archivos XML. Hibernate 3 también guarda cercanía con la especificación *EJB* 3.0 (aunque apareciera antes de la publicación de dicha especificación por *Java Community Process*) y actúa como la espina dorsal de la implementación de EJB 3.0 en *JBoss*.

2.7 JUSTIFICACIÓN DE TECNOLOGÍAS

1) Android

Alternativas: iOS, Windows Phone, Symbian, Bada.

Se ha optado por utilizar Android debido a la gran aceptación de este sistema en los smartphones de todo el mundo, siendo a día de hoy, el sistema operativo para móviles de mayor crecimiento, además de su cada vez mayor utilización en el mundo de los tablets.

Además, otro motivo suficientemente importante como para emplear Android ha sido la facilidad para el desarrollo, suponiendo el hecho de que se programe en Java, lenguaje que se ha utilizado de manera mayoritaria durante carrera. Además, la existencia de un *plugin* de desarrollo para Eclipse, hace incrementar la productividad en la fase de desarrollo.

2) Eclipse

Alternativas: Netbeans.

Se ha optado por la utilización de Eclipse como IDE para el proyecto debido al hecho de que se posee un conocimiento lo suficientemente amplio sobre su funcionamiento como para que la fase de adaptación al entorno de desarrollo no tenga apenas impacto en el proyecto.

Además, como se ha comentado anteriormente, la existencia de un *plugin* para desarrollo en Android hace la elección de Eclipse casi obligada.

3) JSON

Alternativas: XML[12].

A pesar del mayor conocimiento previo de XML, se ha optado por la utilización de JSON debido, en primer lugar, a su facilidad de aprendizaje, por lo que no iba a suponer un gran impacto respecto al XML.

Además, con JSON la información transmitida se reduce considerablemente. Esto es un punto a su favor debido a que se van a utilizar smartphones como plataforma de uso final, y cuanto menor volumen de datos se recuperen desde el servidor, más rápida será la respuesta de la aplicación y más satisfactoria la experiencia para el usuario.

Por último, comentar que con la librería *GSON*[13], la serialización y deserialización de contenido en JSON puede resultar más fácil de lo previsto inicialmente, excepto en un aspecto que se comentará más adelante, en el apartado de decisiones de diseño y detalles de implementación.

4) REST

Alternativas: SOAP.

Se ha decidido utilizar los servicios Web REST debido principalmente al soporte proporcionado por la herramienta Eclipse, la mayor facilidad de desarrollo y sobre todo, la simplicidad en el acceso a los recursos por este sistema.

5) MySQL

Alternativas: Oracle, SQLServer.

La utilización de este gestor de BBDD viene determinada principalmente por experiencias anteriores, en las que se comprobó la facilidad de desarrollo y acceso a la base de datos desde el código Java.

Además, hay que tener en cuenta que la base de datos creada para este Proyecto Fin de Carrera, junto con los servicios Web, no son más que un entorno

de ejemplo, y no pretende ser muy completo ni manejar una gran concurrencia de usuarios. El entorno de explotación final vendría determinado por la integración de la aplicación mediante servicios Web con la base de datos existente en la empresa o corporación que decidiera utilizar mi aplicación.

6) Hibernate

Alternativas: JPA, acceso directo SQL mediante conector JDBC.

Al igual que en otras tecnologías, la utilización de Hibernate para acceder a la capa de persistencia se ha decidido por el conocimiento anterior de esta tecnología, además del soporte de la misma por parte de Eclipse, lo que ha supuesto un ahorro de recursos en este sentido.

Finalmente, comentar que resulta más sencillo realizar acceso directo a la BBDD mediante conector, pero se ha considerado una solución poco elegante y no reutilizable.

3. ANÁLISIS DEL SISTEMA

3.1 DESCRIPCIÓN GENERAL

El sistema desarrollado tiene por objetivo primordial controlar todos los aspectos relacionados con la gestión de inventario. Debido a que la aplicación se va a ejecutar en dispositivos móviles o "smartphones", en distintas localizaciones y frecuentemente, es esencial que la aplicación sea rápida en el manejo de datos, a la vez teniendo la restricción de la poca capacidad de almacenamiento de los dispositivos. Además, el objetivo final de la aplicación es integrarla con las bases de datos de inventario existentes en las distintas organizaciones que utilicen el sistema.

Así pues, se hace necesario contar con una máquina servidor, que disponga de la base de datos de inventario, y que cuente con los mecanismos necesarios para recibir y enviar información desde y hacia los dispositivos móviles que ejecuten la aplicación.

En el siguiente diagrama se puede ver de manera esquemática la estructura principal de todo el sistema:

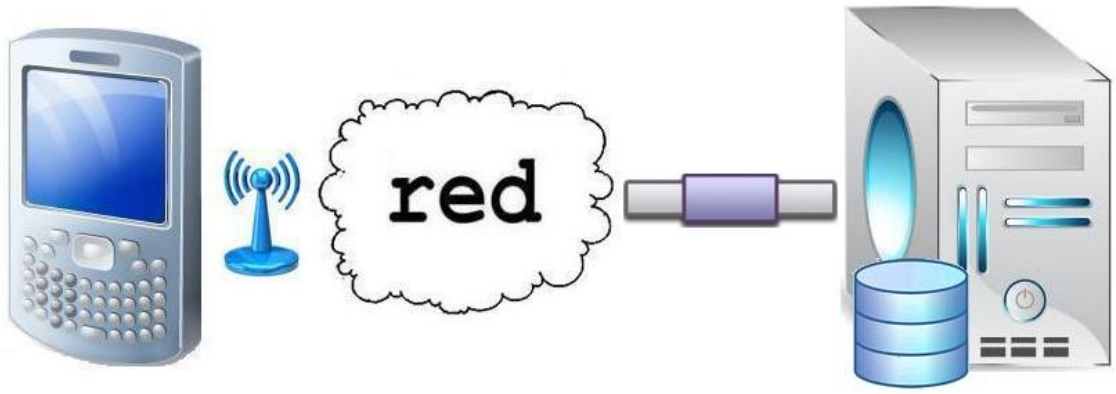


Ilustración 9: Descripción esquemática del sistema

Como se puede ver en la imagen anterior, los dispositivos móviles se comunican con el servidor a través de WiFi, recibiendo éste las comunicaciones por los canales habituales de red. El protocolo de comunicación entre los dos dispositivos será el de los servicios Web.

En la sección de diseño e implementación se describirán en detalle los distintos componentes que hacen posible esta comunicación cliente-servidor.

3.2 CASOS DE USO

A continuación se presenta el diagrama de casos de uso:

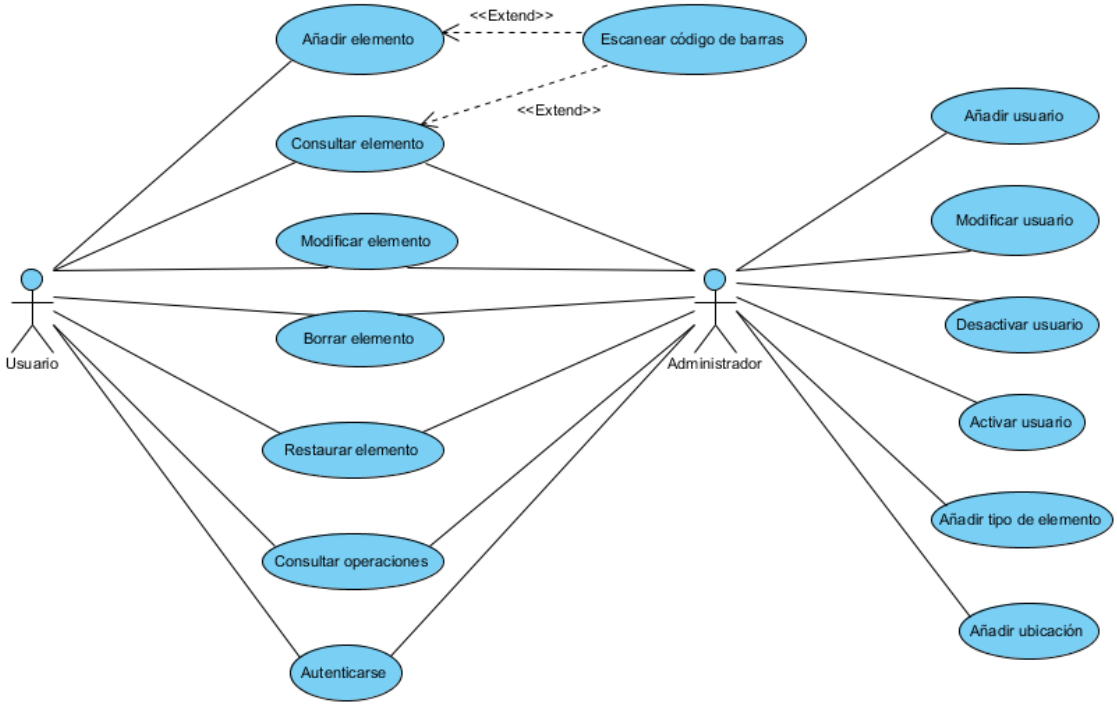


Ilustración 10: Diagrama de casos de uso

Una vez especificados los casos de uso en el diagrama anterior, se muestra una especificación más detallada de éstos mediante la utilización de unas tablas preconcebidas que contienen campos referidos a aspectos o características de la naturaleza del caso de uso. La explicación del significado de cada uno de estos campos se describe a continuación:

- **Id:** Identificador asociado a cada caso de uso, éste deberá ser unívoco y su nomenclatura vendrá dada por CU-XX, siendo XX un número incremental único dentro de los casos de uso cuyo comienzo será 01.

- **Caso de Uso:** Nombre descriptivo del caso de uso.

- **Actores:** Roles de usuario implicados en el caso de uso que tienen la opción de realizarlo.

- **Objetivo:** Descripción de la funcionalidad atribuida al caso de uso, y la razón de su existencia.

- **Precondiciones:** Condiciones necesarias que deben cumplirse para poder realizar el caso de uso.

- **Postcondiciones:** Efectos y consecuencias que provoca la ejecución del caso de uso en el sistema.

- **Escenario básico:** Descripción del flujo principal del caso de uso en relación a la interacción entre el actor y el sistema.

- **Escenario alternativo:** uno o varios de los flujos del caso de uso, diferentes al escenario básico, que se pueden llevar a cabo debido a la evaluación de condiciones de error u otros factores.

Id	**CU-01**	**Caso de Uso**	**Autenticarse**
Actores		Usuario, Administrador	
Objetivo		El usuario se autentica en la aplicación mediante la introducción de su identificador y su contraseña.	
Precondiciones		Ninguna.	
Postcondiciones		El usuario está registrado en el sistema.	
Escenario Básico		1) Se ejecuta la aplicación. 2) El usuario introduce su identificador y su contraseña. 3) Se muestra el menú principal correspondiente a su perfil.	
Escenario Alternativo		Se mostrará un mensaje de error si: - Se introduce un identificador de usuario no existente en el sistema. - Se introduce una contraseña errónea. - El usuario introducido se encuentra desactivado.	

Tabla 4: Caso de uso CU-01

Id	CU-02	Caso de Uso	Añadir elemento
Actores		Usuario	
Objetivo		El usuario añade un nuevo elemento en el sistema.	
Precondiciones		El usuario se ha autenticado previamente.	
Postcondiciones		El nuevo elemento se encuentra añadido en el sistema.	
Escenario Básico		1) Se introduce la información del nuevo elemento (para el código se puede utilizar el escáner). 2) Se confirma la inserción del elemento. 3) La aplicación informa del éxito de la operación.	
Escenario Alternativo		Se mostrará un mensaje de error si: - Se produce un error de BBDD al insertar el elemento. - Se introduce un código inválido en el formulario. - No se selecciona ningún tipo de elemento. - No se introduce una descripción. - No se introduce la ubicación completa.	

Tabla 5: Caso de uso CU-02

Id	CU-03	Caso de Uso	Consultar elemento
Actores		Usuario, Administrador	
Objetivo		El usuario realiza una operación de consulta de elementos en el sistema.	
Precondiciones		El usuario se ha autenticado previamente.	
Postcondiciones		Ninguna, no se producen alteraciones en el sistema.	
Escenario Básico		1) Se introducen las condiciones de búsqueda en el formulario. 2) Se confirma la búsqueda. 3) Se muestra la lista de elementos que cumplen los criterios de búsqueda recuperados del sistema.	

Escenario Alternativo	Se mostrará un mensaje de error si: - No se introduce ningún criterio de búsqueda (no se pueden mostrar todos los elementos). - Se produce un error en la recuperación de la información desde el servidor.

Tabla 6: Caso de uso CU-03

Id	**CU-04**	**Caso de Uso**	**Escanear código de barras**
Actores		Usuario, Administrador	
Objetivo		El usuario escanea un código de barras de un elemento de inventario.	
Precondiciones		Se ha iniciado una nueva inserción de elemento o una nueva consulta.	
Postcondiciones		Se completa la inserción o la consulta.	
Escenario Básico		1) Se escanea el código. 2) A: si se trata de una inserción, se guarda el código escaneado en el formulario. B: si se trata de una consulta, se ejecuta la consulta con el código escaneado.	
Escenario Alternativo		Se mostrará un mensaje de error si: - Se produce un error en el escaneo del código.	

Tabla 7: Caso de uso CU-04

Id	**CU-05**	**Caso de Uso**	**Modificar elemento**
Actores		Usuario, Administrador	
Objetivo		El usuario modifica la información de un elemento ya existente en el sistema.	
Precondiciones		Se ha ejecutado una consulta de elementos previa.	
Postcondiciones		El elemento se actualiza en el sistema con la nueva información.	
Escenario Básico		1) Se modifica la información existente del	

	elemento en el formulario. 2) Se guarda el elemento con la nueva información. 3) Se muestra un mensaje informando del éxito de la operación.
Escenario Alternativo	Se mostrará un mensaje de error si: - Se introduce un código de elemento inválido. - No se selecciona ningún tipo de elemento. - No se introduce ninguna descripción. - No se selecciona la ubicación completa.

Tabla 8: Caso de uso CU-05

Id	**CU-06**	**Caso de Uso**	**Borrar elemento**
Actores		Usuario, Administrador	
Objetivo		El usuario borra un elemento existente en el sistema.	
Precondiciones		Se ha ejecutado una consulta de elementos previa.	
Postcondiciones		El elemento se guarda en el sistema con el estado actualizado.	
Escenario Básico		1) Se pulsa el botón de borrado en el formulario. 2) Se muestra un mensaje de éxito de la operación y se actualiza el formulario cambiando el botón para activaciones posteriores.	
Escenario Alternativo		Se mostrará un mensaje de error si: - Se produce un error en la actualización del elemento en el servidor.	

Tabla 9: Caso de uso CU-06

Id	**CU-07**	**Caso de Uso**	**Restaurar elemento**
Actores		Usuario, Administrador	
Objetivo		El usuario restaura un elemento existente en el sistema.	

Precondiciones	Se ha ejecutado una consulta de elementos previa.
Postcondiciones	El elemento se guarda en el sistema con el estado actualizado.
Escenario Básico	1) Se pulsa el botón de restauración en el formulario. 2) Se muestra un mensaje de éxito de la operación y se actualiza el formulario cambiando el botón para borrados posteriores.
Escenario Alternativo	Se mostrará un mensaje de error si: - Se produce un error en la actualización del elemento en el servidor.

Tabla 10: Caso de uso CU-07

Id	**CU-08**	**Caso de Uso**	**Consultar operaciones**
Actores	Usuario, Administrador		
Objetivo	El usuario realiza una consulta de operaciones en el sistema.		
Precondiciones	El usuario se ha autenticado en el sistema previamente.		
Postcondiciones	Ninguna, no se realizan cambios en el sistema.		
Escenario Básico	1) Se introducen las condiciones de búsqueda en el formulario. 2) Se confirma la búsqueda. 3) Se muestra la lista de operaciones que cumplen los criterios de búsqueda recuperados del sistema.		
Escenario Alternativo	Se mostrará un mensaje de error si: - La fecha de fin de rango es mayor que la fecha de inicio. - Se produce un error en la recuperación de la información desde el servidor.		

Tabla 11: Caso de uso CU-08

Id	CU-09	Caso de Uso	Añadir usuario
Actores		Administrador	
Objetivo		El usuario añade un nuevo usuario en el sistema.	
Precondiciones		El usuario se ha autenticado en el sistema previamente.	
Postcondiciones		El nuevo usuario se encuentra en el sistema.	
Escenario Básico		1) Se introduce la información del nuevo usuario en el formulario. 2) Se confirma la inserción del nuevo usuario. 3) Se muestra un mensaje de éxito de la operación.	
Escenario Alternativo		Se mostrará un mensaje de error si: - No se introduce un identificador de usuario. - El identificador de usuario introducido ya se encuentra en uso. - No se completa alguno de los dos campos de contraseña. - La información de los dos campos de contraseña no coincide. - No se selecciona un perfil de usuario. - Se produce un error al guardar el usuario en el servidor.	

Tabla 12: Caso de uso CU-09

Id	CU-10	Caso de Uso	Modificar usuario
Actores		Administrador	
Objetivo		El usuario modifica la información de un usuario existente en el sistema.	
Precondiciones		Se muestra en detalle el usuario tras haberlo seleccionado de la lista de todos los usuarios existentes.	
Postcondiciones		Se actualiza el usuario con la nueva información.	
Escenario Básico		1) Se modifica la información del usuario en el formulario. 2) Se pulsa el botón de guardar el usuario. 3) Se muestra un mensaje de éxito de la	

	operación.
Escenario Alternativo	Se mostrará un mensaje de error si: - No se completa alguno de los dos campos de contraseña. - La información de los dos campos de contraseña no coincide. - No se selecciona un perfil de usuario. - Se produce un error al guardar el usuario en el servidor.

Tabla 13: Caso de uso CU-10

Id	**CU-11**	**Caso de Uso**	**Desactivar usuario**
Actores		Administrador	
Objetivo		El usuario desactiva un usuario previamente existente en el sistema.	
Precondiciones		Se muestra en detalle el usuario tras haberlo seleccionado de la lista de todos los usuarios existentes.	
Postcondiciones		Se guarda el usuario con el estado actualizado.	
Escenario Básico		1) Se pulsa el botón de desactivar el usuario en el formulario. 2) Se muestra un mensaje de éxito de la operación, y se actualiza el formulario cambiando el botón para futuras activaciones.	
Escenario Alternativo		Se mostrará un mensaje de error si: - Se produce un error en la comunicación con el servidor.	

Tabla 14: Caso de uso CU-11

Id	**CU-12**	**Caso de Uso**	**Activar usuario**
Actores		Administrador	
Objetivo		El usuario activa un usuario previamente existente en el sistema.	

Precondiciones	Se muestra en detalle el usuario tras haberlo seleccionado de la lista de todos los usuarios existentes.
Postcondiciones	Se guarda el usuario con el estado actualizado.
Escenario Básico	1) Se pulsa el botón de activar el usuario en el formulario. 2) Se muestra un mensaje de éxito de la operación, y se actualiza el formulario cambiando el botón para futuras desactivaciones.
Escenario Alternativo	Se mostrará un mensaje de error si: - Se produce un error en la comunicación con el servidor.

Tabla 15: Caso de uso CU-12

Id	**CU-13**	**Caso de Uso**	**Añadir tipo de elemento**
Actores		Administrador	
Objetivo		El usuario añade un nuevo tipo de elemento al sistema.	
Precondiciones		El usuario se ha autenticado en el sistema previamente.	
Postcondiciones		El nuevo tipo de elemento se guarda en el sistema.	
Escenario Básico		1) Se introduce el nombre del nuevo tipo de elemento. 2) Se muestra un mensaje de éxito de la operación, y se actualiza la lista de tipos de elemento para mostrar el que se acaba de añadir.	
Escenario Alternativo		Se mostrará un mensaje de error si: - Se introduce un tipo de elemento ya existente. - Se produce un error en la comunicación con el servidor.	

Tabla 16: Caso de uso CU-13

Id	**CU-14**	**Caso de Uso**	**Añadir ubicación**

Actores	Administrador
Objetivo	El usuario añade una nueva ubicación en el sistema.
Precondiciones	El usuario se ha autenticado en el sistema previamente.
Postcondiciones	La nueva ubicación se guarda en el sistema.
Escenario Básico	1) Se introduce la información de la nueva ubicación. 2) Se pulsa el botón para añadir la nueva ubicación al sistema. 3) Se muestra un mensaje de éxito de la operación.
Escenario Alternativo	Se mostrará un mensaje de error si: - No se introduce toda la información de la nueva ubicación. - Se produce un error en la comunicación con el servidor.

Tabla 17: Caso de uso CU-14

3.3 REQUISITOS DEL SISTEMA

Para realizar la definición de los requisitos se usará la siguiente plantilla en forma de tabla, cuyos campos serán explicados después:

Identificador		**Versión**	
Nombre			
Descripción			
Prioridad		**Fuente**	

Tabla 18: Prototipo de tabla para la definición de requisitos

El significado de los campos de la tabla es el siguiente:

- Identificador: Código unívoco que define al requisito. El prototipo de identificador será el siguiente: RX-YZ, siendo X el tipo de categoría de requisito (F si se trata de funcional, NF si se trata de no funcional, R si es de restricción), e YZ números que irán desde "01" hasta "99", tantos como se necesiten para cubrir todos los requisitos existentes.

- Versión: Será un número, comenzando por "1", que definirá el número de versión del requisito. Cada vez que éste cambie su definición o funcionalidad, el número de versión se incrementará en una unidad.

- Nombre: Cadena corta que describe de manera rápida la funcionalidad y/o ámbito del requisito en cuestión.

- Descripción: Definición más amplia de las características del requisito, tratando aspectos más concretos como objetivos, usuarios, restricciones, etc.

- Prioridad: Grado de urgencia con el que debe cumplirse el requisito. Puede ser Alta, Media o Baja.

- Fuente: Indica la procedencia de la necesidad del requisito.

3.3.1 REQUISITOS FUNCIONALES

Identificador	RF-01	**Versión**	1
Nombre	Autenticación		
Descripción	Para acceder al sistema se deberá proporcionar la información de autenticación. Si la información es correcta, se mostrará el menú principal de la aplicación del perfil correspondiente; en caso contrario, se informará del error en pantalla.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 19: Requisito funcional RF-01

Identificador	RF-02	Versión	1
Nombre	Añadir elementos		
Descripción	El sistema permitirá añadir nuevos elementos al inventario, completando la información de dicho elemento a través de la interfaz. Esta operación sólo la podrá realizar el perfil usuario.		
Prioridad	Alta	Fuente	Descripción del proyecto.

Tabla 20: Requisito funcional RF-02

Identificador	RF-03	Versión	1
Nombre	Consultar elementos		
Descripción	El sistema permitirá realizar consultas sobre elementos, según la siguiente información: - Por código, pudiendo introducirse éste manualmente o leyéndolo a través de la cámara[14]. - Por localización del elemento. - Por tipo de elemento.		
Prioridad	Alta	Fuente	Descripción del proyecto.

Tabla 21: Requisito funcional RF-03

Identificador	RF-04	Versión	1
Nombre	Modificar elementos		
Descripción	El sistema permitirá modificar la información de elementos consultados con anterioridad.		
Prioridad	Alta	Fuente	Descripción del proyecto.

Tabla 22: Requisito funcional RF-04

Identificador	RF-05	**Versión**	1
Nombre	Eliminar elementos		
Descripción	El sistema permitirá eliminar elementos del inventario, aunque éstos no se borrarán realmente, sino que se marcarán como borrados, y se conservarán para futuras consultas.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 23: Requisito funcional RF-05

Identificador	RF-06	**Versión**	1
Nombre	Restaurar elementos		
Descripción	El sistema permitirá restaurar elementos que previamente hayan sido marcados como borrados.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 24: Requisito funcional RF-06

Identificador	RF-07	**Versión**	1
Nombre	Consultar operaciones		
Descripción	El sistema permitirá consultar ciertas operaciones sobre inventario, tales como: - Operaciones realizadas por un determinado usuario. - Operaciones realizadas sobre un determinado elemento. - Operaciones de un determinado tipo (creación, edición, borrado o restauración). - Operaciones realizadas en un determinado rango de fechas. Los filtros expuestos anteriormente podrán ser combinados. El perfil usuario solo podrá ver		

	operaciones realizadas por él mismo.		
Prioridad	Media	**Fuente**	Descripción del proyecto.

Tabla 25: Requisito funcional RF-07

Identificador	RF-08	**Versión**	1
Nombre	Añadir usuarios		
Descripción	El sistema permitirá añadir nuevos usuarios al sistema, completando la información de dicho usuario a través de la interfaz. Esta operación sólo la podrá realizar el perfil administrador.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 26: Requisito funcional RF-08

Identificador	RF-09	**Versión**	1
Nombre	Modificar usuarios		
Descripción	El sistema permitirá modificar la información de los usuarios añadidos anteriormente, excepto el identificador, que será único e invariable. Esta operación sólo la podrá realizar el perfil administrador.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 27: Requisito funcional RF-09

Identificador	RF-10	**Versión**	1
Nombre	Desactivar usuarios		
Descripción	El sistema permitirá dar de baja a cualquier usuario añadido con anterioridad, negándole de este modo el acceso futuro a la aplicación. Esta operación sólo la		

	podrá realizar el perfil administrador.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 28: Requisito funcional RF-10

Identificador	RF-11	**Versión**	1
Nombre	Activar usuarios		
Descripción	El sistema permitirá activar a cualquier usuario desactivado con anterioridad, permitiéndole de este modo el acceso a la aplicación de nuevo. Esta operación sólo la podrá realizar el perfil administrador.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 29: Requisito funcional RF-11

Identificador	RF-12	**Versión**	1
Nombre	Añadir tipos de elemento		
Descripción	El sistema permitirá añadir nuevos tipos de elemento al sistema, que luego se utilizarán para clasificar los elementos que se añadan con posterioridad. Esta operación sólo la podrá realizar el perfil administrador.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 30: Requisito funcional RF-12

Identificador	RF-13	**Versión**	1
Nombre	Añadir ubicaciones		
Descripción	El sistema permitirá añadir nuevas ubicaciones al sistema. Estas ubicaciones podrán ser completamente		

	nuevas, o bien a partir de cierto nivel de especificidad ya existente. Esta operación sólo la podrá realizar el perfil administrador.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 31: Requisito funcional RF-13

3.3.2 REQUISITOS NO FUNCIONALES

Identificador	RNF-01	**Versión**	1
Nombre	Resolución de la aplicación		
Descripción	La aplicación se adaptará a la resolución por defecto del dispositivo en que se ejecute.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 32: Requisito no funcional RNF-01

Identificador	RNF-02	**Versión**	1
Nombre	Licencia GPL		
Descripción	El código completo de la aplicación será de libre distribución, gozando de licencia GNU General Public License.		
Prioridad	Media	**Fuente**	Descripción del proyecto.

Tabla 33: Requisito no funcional RNF-02

Identificador	RNF-03	**Versión**	1
Nombre	Perfiles de usuario		

Descripción	En el sistema existirán los siguientes perfiles de usuario: - Administrador. - Usuario. Un usuario cualquiera pertenecerá a uno y solo uno de los perfiles anteriores.		
Prioridad	Alta	**Fuente**	Decisión de diseño.

Tabla 34: Requisito no funcional RNF-03

Identificador	RNF-04	**Versión**	1
Nombre	Interfaz de usuario		
Descripción	La interfaz de usuario de la aplicación móvil será poco cargada y fácil de manejar, siendo un aspecto fundamental su alta usabilidad.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 35: Requisito no funcional RNF-04

Identificador	RNF-05	**Versión**	1
Nombre	Identificación de usuarios		
Descripción	El atributo ID de usuario será la única forma de identificar a un usuario en el sistema.		
Prioridad	Media	**Fuente**	Decisión de diseño.

Tabla 36: Requisito no funcional RNF-05

Identificador	RNF-06	**Versión**	1
Nombre	Activación/desactivación de elementos y usuarios		

Descripción	Cuando se activa o desactiva un elemento o un usuario, la operación es prioritaria y se deberá realizar de manera inmediata.		
Prioridad	Media	**Fuente**	Decisión de diseño.

Tabla 37: Requisito no funcional RNF-06

Identificador	RNF-07	**Versión**	1
Nombre	Minimización de información introducida		
Descripción	Se minimizará la inserción de información ya existente en el sistema por parte del usuario.		
Prioridad	Baja	**Fuente**	Decisión de diseño.

Tabla 38: Requisito no funcional RNF-07

3.3.3 REQUISITOS DE RESTRICCIÓN

Identificador	RR-01	**Versión**	1
Nombre	Versión de Android		
Descripción	El sistema será ejecutable exclusivamente en el sistema operativo Android, versión 2.1 o superior.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 39: Requisito de restricción RR-01

Identificador	RR-02	**Versión**	1
Nombre	Barcode Scanner		
Descripción	Será necesario tener instalado en el dispositivo el programa Barcode Scanner, para ejecutar la		

	funcionalidad de lectura de códigos.		
Prioridad	Media	**Fuente**	Decisión de diseño.

Tabla 40: Requisito de restricción RR-02

Identificador	RR-03	**Versión**	1
Nombre	Cámara de fotos		
Descripción	El dispositivo en el que se ejecute la aplicación deberá contar con cámara de fotos.		
Prioridad	Media	**Fuente**	Decisión de diseño.

Tabla 41: Requisito de restricción RR-03

Identificador	RR-04	**Versión**	1
Nombre	Guardado de contraseñas		
Descripción	Las contraseñas se guardarán en la BBDD después de haberlas aplicado una función Hash, para evitar su recuperación.		
Prioridad	Alta	**Fuente**	Decisión de diseño.

Tabla 42: Requisito de restricción RR-04

Identificador	RR-05	**Versión**	1
Nombre	Conexión a Internet		
Descripción	Todo dispositivo que ejecute la aplicación deberá contar con una conexión funcional a Internet, para poder realizar las comunicaciones con el servidor.		
Prioridad	Alta	**Fuente**	Descripción del proyecto.

Tabla 43: Requisito de restricción RR-05

3.4 MATRIZ DE TRAZABILIDAD CASOS DE USO – REQUISITOS

CU \ RU	RF-01	RF-02	RF-03	RF-04	RF-05	RF-06	RF-07	RF-08	RF-09	RF-10	RF-11	RF-12	RF-13
CU-01	X												
CU-02		X											
CU-03			X										
CU-04		X	X										
CU-05				X									
CU-06					X								
CU-07						X							
CU-08							X						
CU-09								X					
CU-10									X				
CU-11										X			
CU-12											X		
CU-13												X	
CU-14													X

Tabla 44: Matriz de trazabilidad casos de uso - requisitos

4. DISEÑO E IMPLEMENTACIÓN

4.1 ARQUITECTURA DEL SISTEMA

Al igual que gran parte de las aplicaciones cliente-servidor, este sistema sigue un patrón *MVC* (Modelo-Vista-Controlador)[15]. Teóricamente, los componentes de este patrón tienen la siguiente función:

- El **modelo** tiene la función de acceder a la capa de datos, independientemente del sistema de almacenamiento empleado.
- El **controlador** tiene la función de gestionar los eventos y la lógica de control, así como aplicar la lógica de negocio.
- La **vista** tiene la función de recibir los datos del modelo a través del controlador y mostrarlos mediante una interfaz de usuario.

Ahora bien, en el ámbito de este sistema, el patrón se aplica de la siguiente manera:

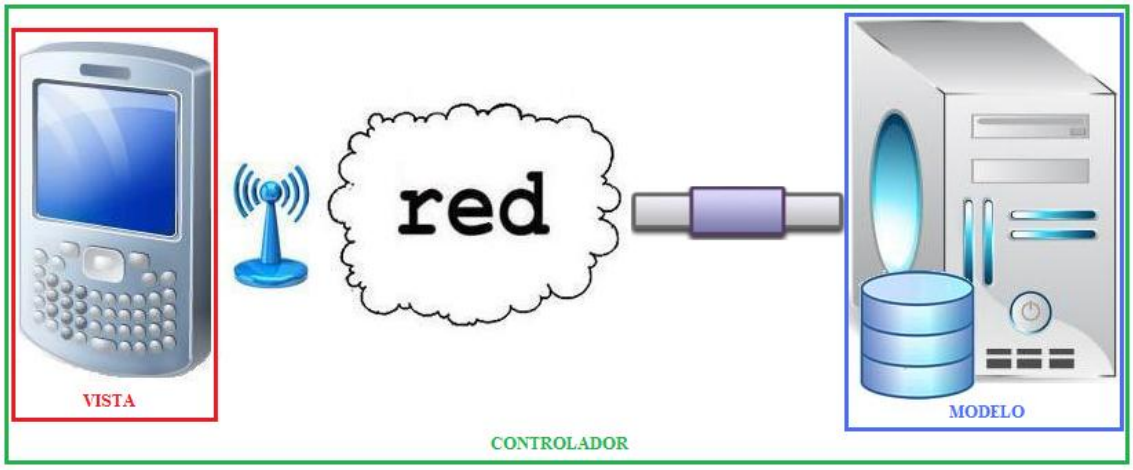


Ilustración 11: Modelo-vista-controlador en el sistema

En cuanto al modelo y a la vista, no hay nada especial; la máquina servidor actúa como modelo, albergando el sistema gestor de bases de datos, y el dispositivo móvil

hace las veces de vista, ofreciendo al usuario la interfaz gráfica para interactuar con el sistema.

Lo especial del patrón en este sistema es el componente controlador, que lo comparten tanto la máquina servidor como el dispositivo móvil. Esto es así porque ambos gestionan parte de la lógica de negocio; la máquina servidor maneja todo lo relacionado con el tratamiento de los datos del modelo, además de la comunicación con el dispositivo móvil a través de los servicios Web. El dispositivo móvil, por su parte, alberga una parte de la lógica de negocio, en este caso la de la aplicación móvil, que contiene no sólo la lógica propia del funcionamiento de una aplicación (navegación entre pantallas, formateo de datos, tratamiento de errores), sino que también es responsable del envío y recepción de datos al servidor, y debe serializar y deserializar los datos en formato JSON, y realizar las peticiones a los servicios Web.

Una vez explicado todo lo relacionado con la distribución general de la arquitectura de la aplicación, se procede a describir la arquitectura con más detalle. A continuación se muestra el diagrama de despliegue del sistema, que se explicará a continuación:

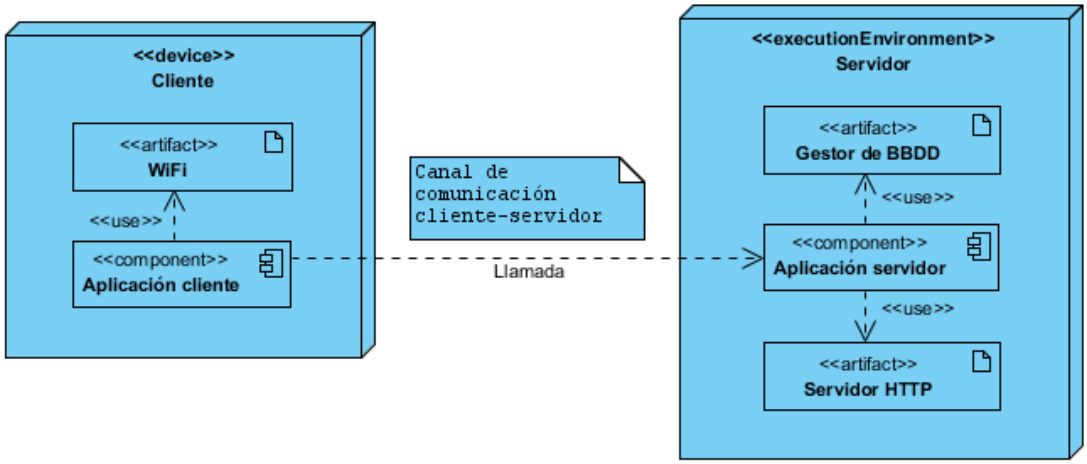


Ilustración 12: Diagrama de despliegue del sistema

Como se puede observar, hay dos dispositivos principales: el dispositivo cliente y el servidor. El dispositivo cliente tiene un componente principal, la aplicación cliente, que hace uso de un artefacto, el WiFi, para realizar las peticiones al servidor. Por su parte, el dispositivo servidor tiene también un componente principal, la aplicación servidor, que utiliza dos artefactos, el gestor de BBDD, para acceder a los datos del

modelo, y el servidor HTTP, que recoge las peticiones realizadas desde el cliente a través de servicios Web.

A continuación se va a profundizar más en los componentes principales de los dispositivos cliente y servidor, la aplicación cliente y la aplicación servidor.

La siguiente imagen se centra en los subcomponentes que forman las dos aplicaciones, así como su interacción entre ellas:

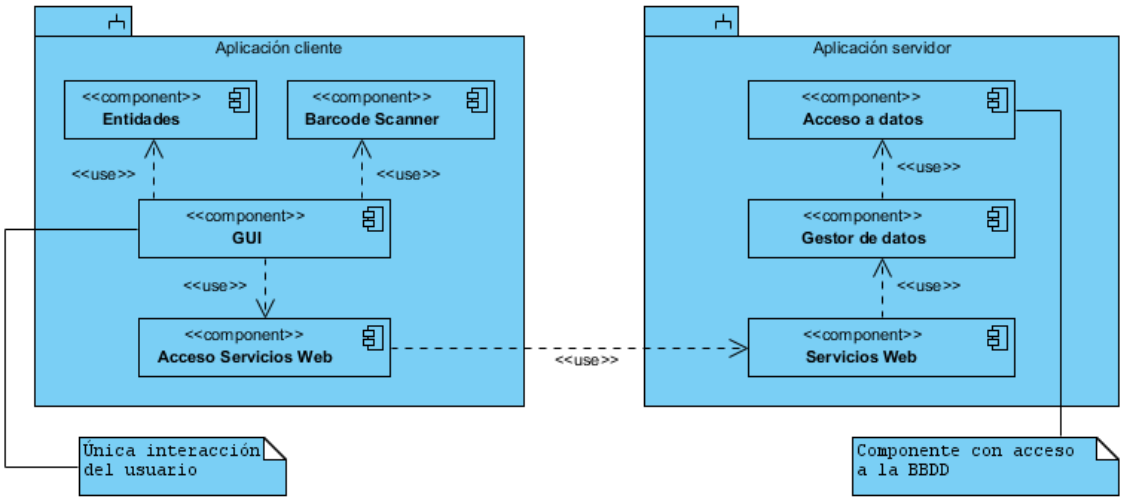


Ilustración 13: Diagrama de componentes del sistema

Para explicar los componentes de la ilustración anterior, se utilizará una representación en forma de tabla, de forma análoga a lo descrito anteriormente en los casos de uso y en los requisitos.

Los campos de las tablas tienen el siguiente significado:

- Identificador: Código unívoco que define al componente. El prototipo de identificador será el siguiente: CX-YZ, siendo X el tipo de categoría de componente (C si se refiere a la aplicación cliente, S si es de la aplicación servidor), e YZ números que irán desde "01" hasta "99", tantos como se necesiten para cubrir todos los componentes.

- Nombre: Nombre del componente, tal y como se muestra en el diagrama de componentes anterior.

- Descripción: Descripción de las características del componente.

- Relaciones: Identificadores de componentes relacionados con el actual.

Identificador	CC-01
Nombre	GUI
Descripción	Es el componente que controla la interfaz de usuario. Contiene todas las clases que son actividades en Android, una por cada pantalla de la interfaz de usuario.
Relaciones	CC-02, CC-03, CC-04

Tabla 45: Componente GUI

Identificador	CC-02
Nombre	Acceso Servicios Web
Descripción	Es el componente que realiza las peticiones a los servicios Web. Contiene la clase que accede a los servicios Web y todas las clases que realizan las tareas de las peticiones.
Relaciones	CC-01, CS-02

Tabla 46: Componente Acceso Servicios Web

Identificador	CC-03
Nombre	Entidades
Descripción	Contiene las clases que modelan los tipos de datos que se manejan en la aplicación, tales como *Elemento, Usuario, Operación…*
Relaciones	CC-01

Tabla 47: Componente Entidades

Identificador	CC-04
Nombre	Barcode Scanner
Descripción	Componente que maneja la lectura de códigos de barras. Contiene las clases proporcionadas por *Barcode Scanner* para integrar la lectura de códigos en las aplicaciones.
Relaciones	CC-01

Tabla 48: Componente Barcode Scanner

Identificador	CS-01
Nombre	Gestor de datos
Descripción	Componente que gestiona el acceso a los datos. Contiene las clases de gestión de cada uno de los tipos de datos manejados en el sistema, correspondientes a cada tabla de la base de datos.
Relaciones	CS-02, CS-03

Tabla 49: Componente Gestor de datos

Identificador	CS-02
Nombre	Servicios Web
Descripción	Componente que gestiona los servicios Web. Contiene las clases correspondientes a las definiciones de los servicios Web de cada tipo de datos manejado.
Relaciones	CS-01, CC-02

Tabla 50: Componente Servicios Web

Identificador	CS-03
Nombre	Acceso a datos

Descripción	Componente que accede directamente a los datos. Contiene las clases que modelan cada uno de los tipos de datos manejados por la aplicación, así como las clases que acceden a la base de datos, a través de Hibernate.
Relaciones	CS-01

Tabla 51: Componente Acceso a datos

4.2 DISEÑO DETALLADO

En esta sección se van a mostrar los diagramas de clases detallados de todo el sistema, con el objetivo de poder ver la aplicación hasta el máximo nivel de detalle.

Como se ha explicado en el punto anterior, el sistema desarrollado consta de dos subsistemas claramente diferenciados, la aplicación cliente y la aplicación servidor. Así pues, los diagramas se van a dividir entre los correspondientes diferenciados.

Dentro de cada subsistema, la estructura a seguir va a ser la siguiente: primero se mostrará el diagrama de clases completo de cada subsistema, sin mostrar los detalles de atributos y métodos, mostrando simplemente las clases y las relaciones entre ellas. Después se mostrará el mismo diagrama dividido en zonas; cada una de las zonas correspondientes a uno de los diagramas mostrados a continuación, ya expuestos con todo lujo de detalles, con todos los atributos y métodos definidos. Siguiendo esta estructura, primero se mostrarán los diagramas correspondientes a la aplicación cliente, y a continuación los correspondientes a la aplicación servidor.

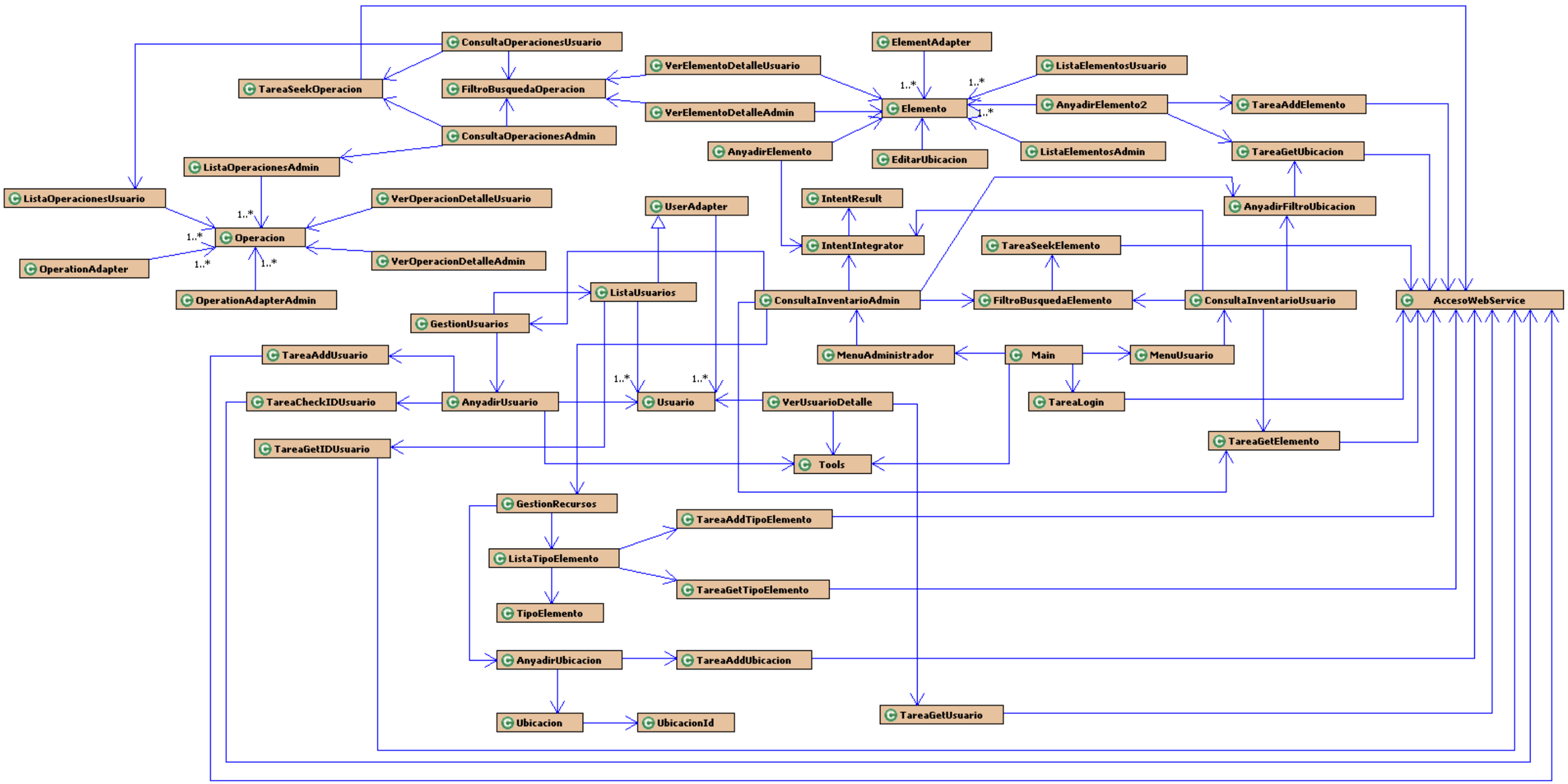


Ilustración 14: Diagrama de clases de la aplicación cliente

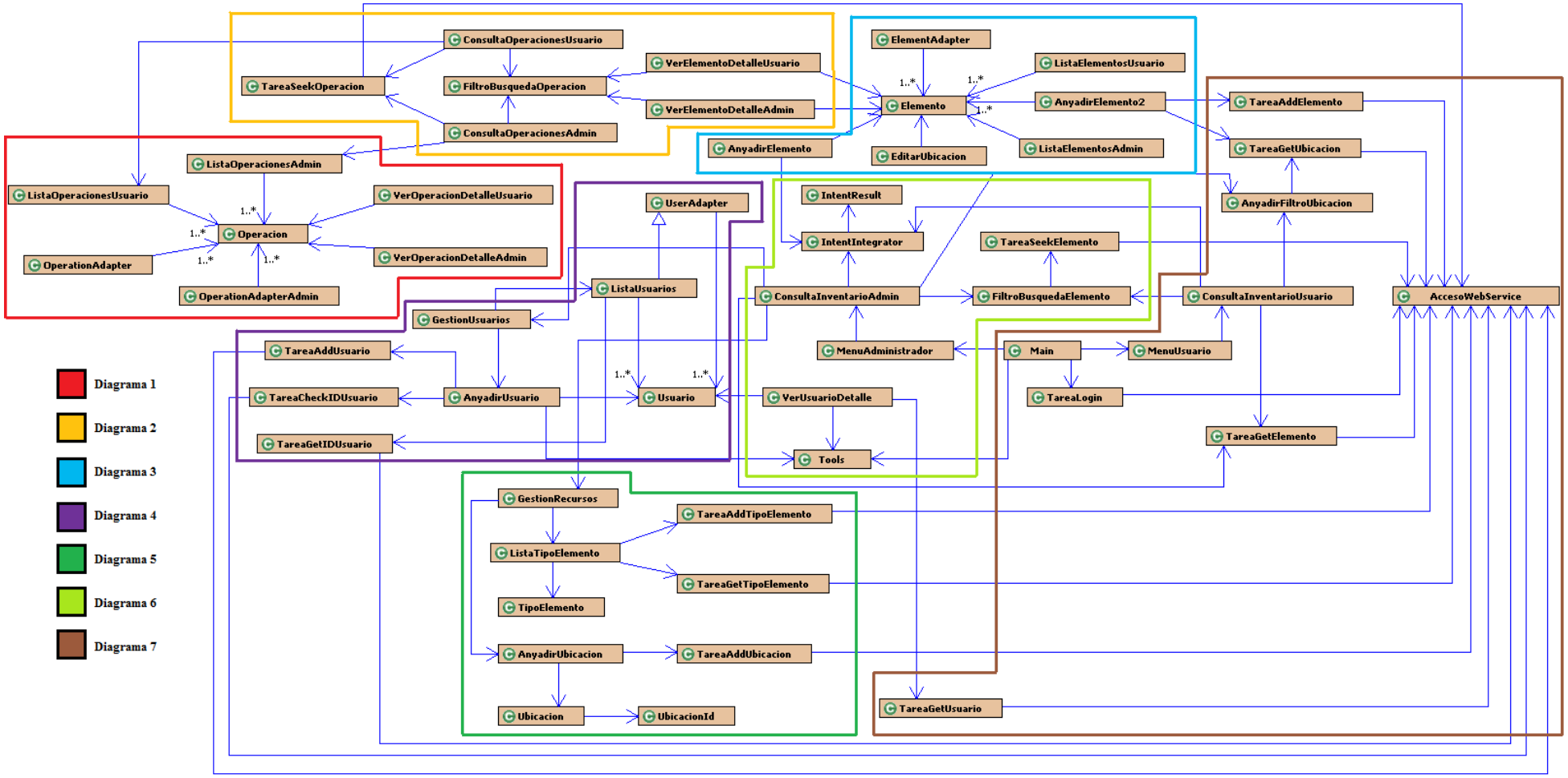


Ilustración 15: Diagrama de clases de la aplicación cliente dividido por zonas

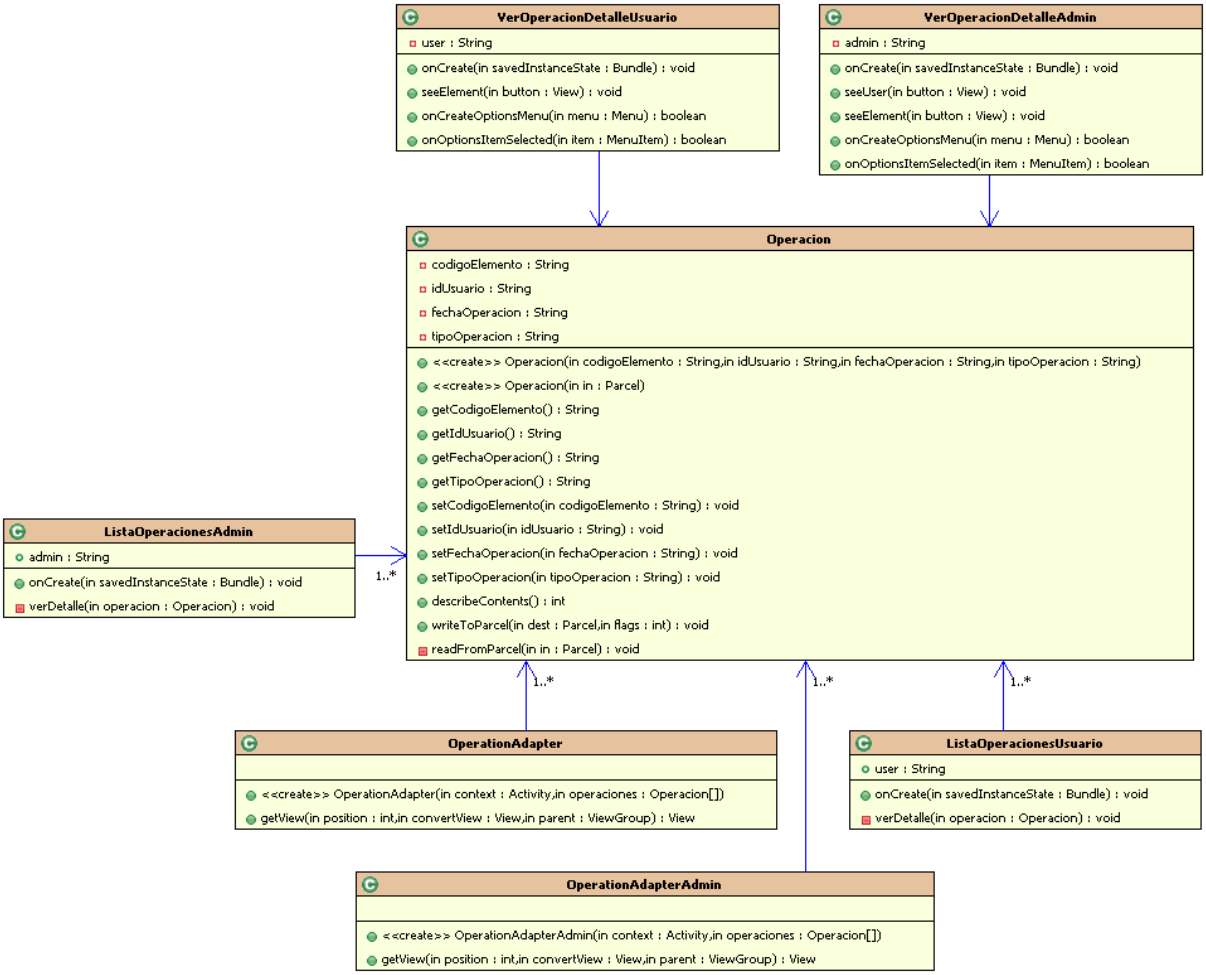


Ilustración 16: Subdiagrama de clases 1 de la aplicación cliente

Ilustración 17: Subdiagrama de clases 2 de la aplicación cliente

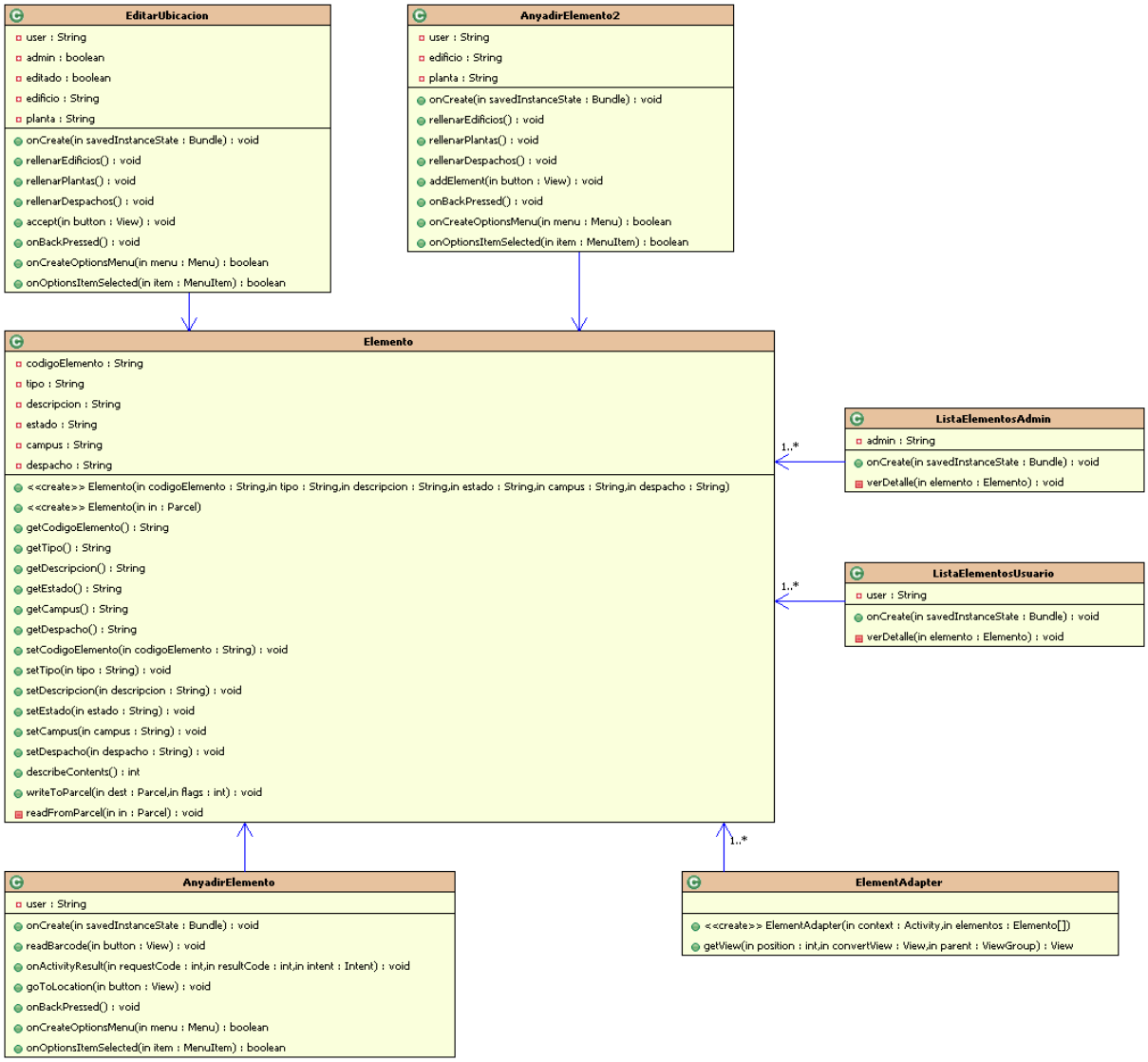


Ilustración 18: Subdiagrama de clases 3 de la aplicación cliente

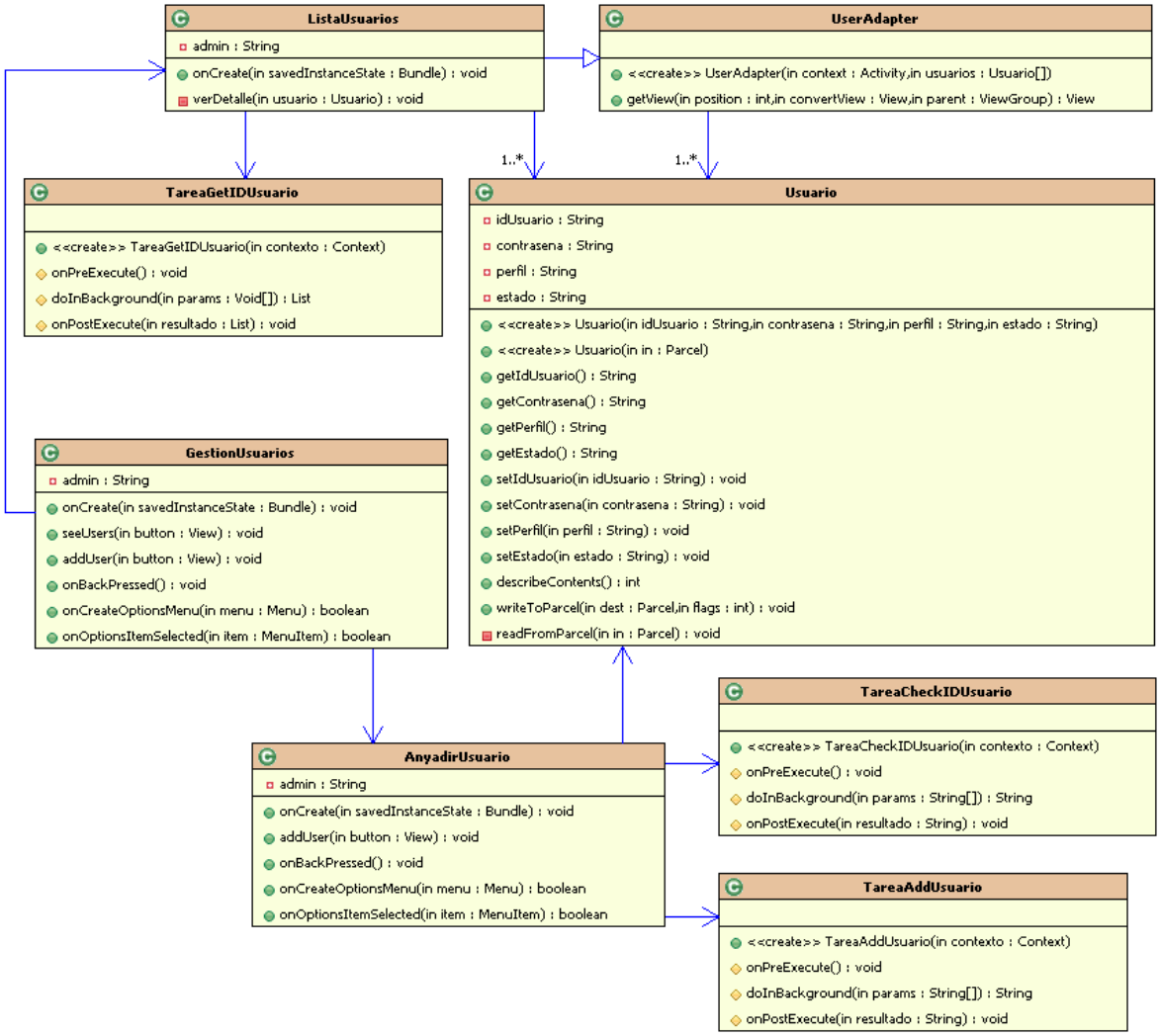


Ilustración 19: Subdiagrama de clases 4 de la aplicación cliente

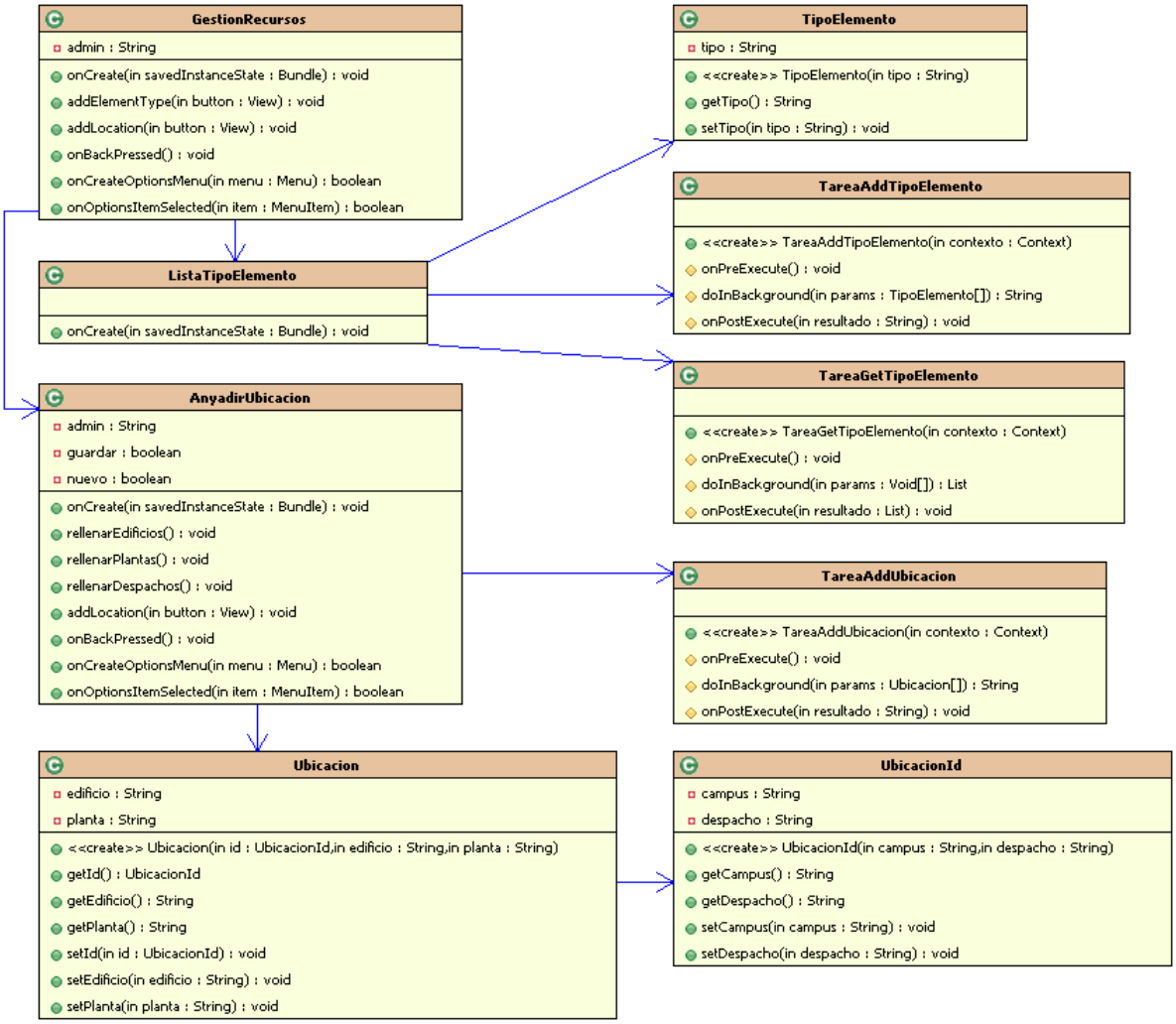


Ilustración 20: Subdiagrama de clases 5 de la aplicación cliente

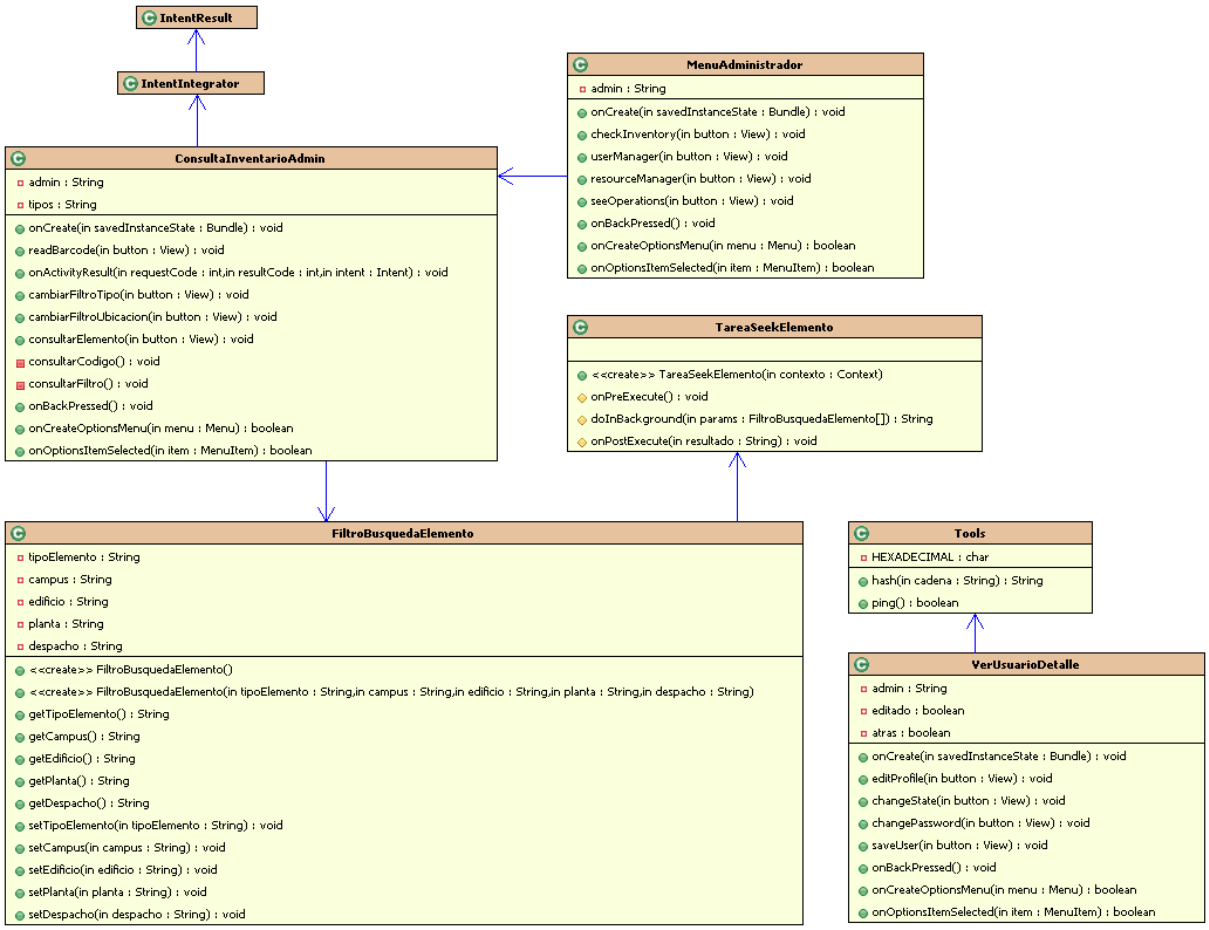


Ilustración 21: Subdiagrama de clases 6 de la aplicación cliente

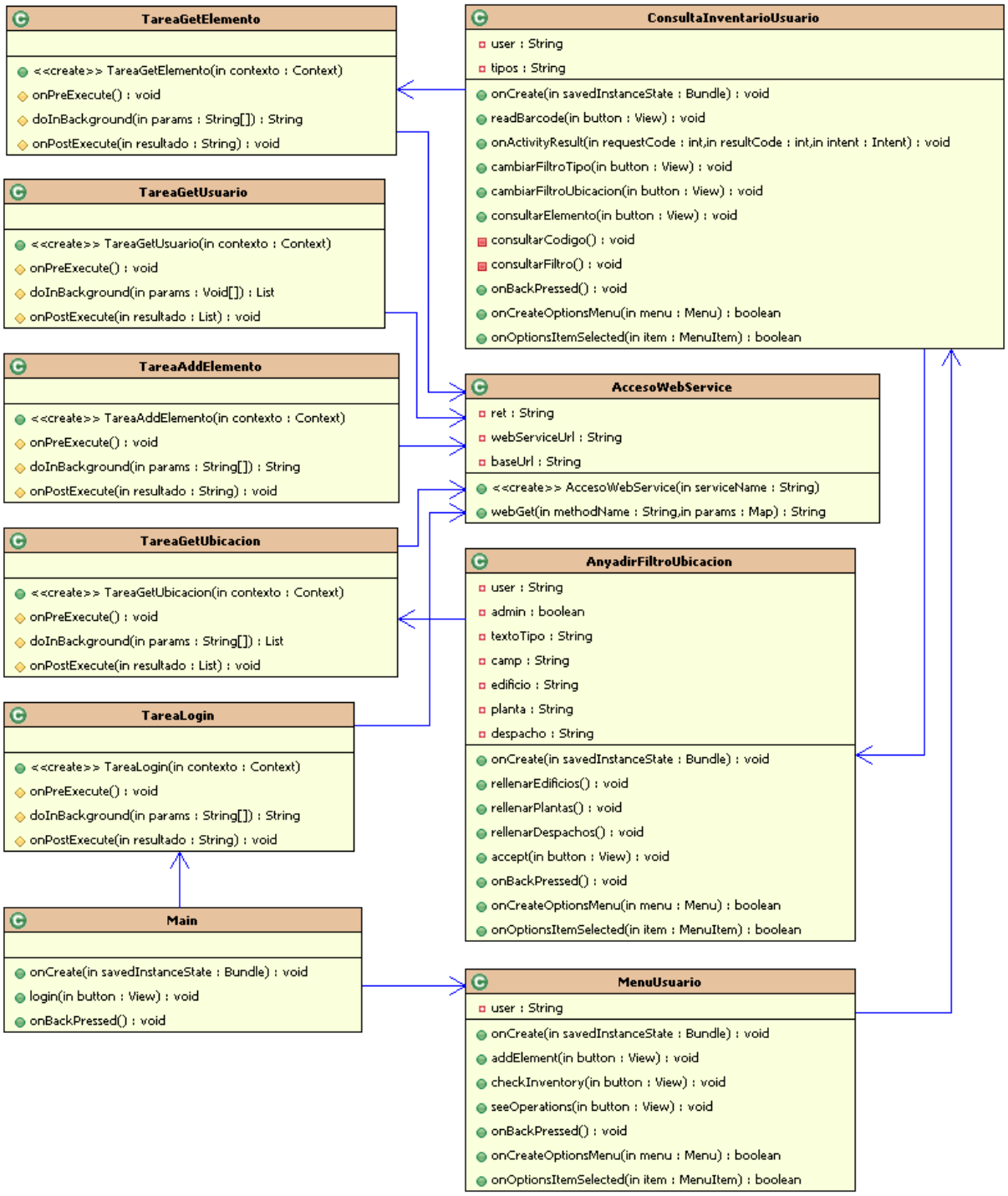


Ilustración 22: Subdiagrama de clases 7 de la aplicación cliente

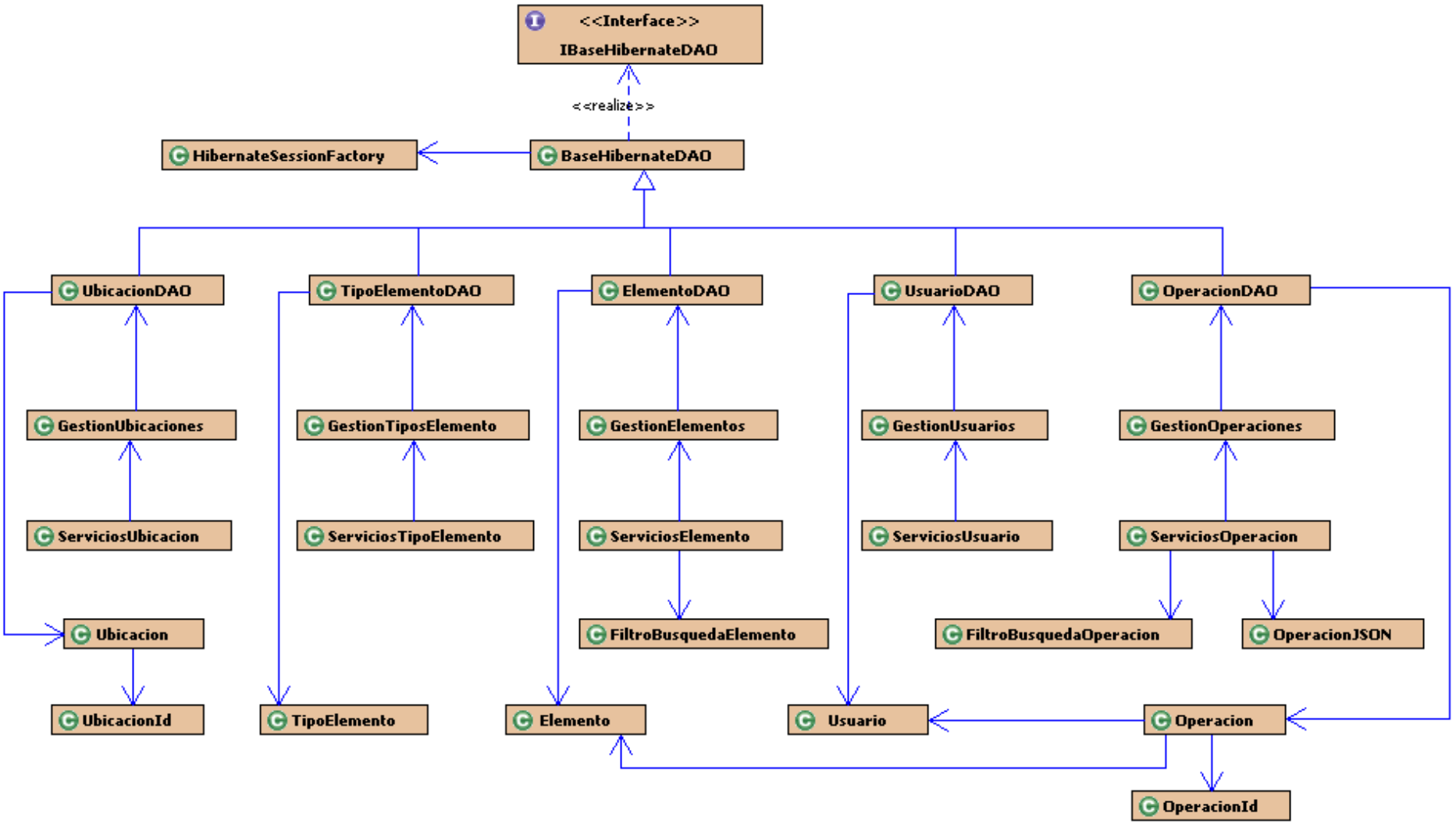


Ilustración 23: Diagrama de clases de la aplicación servidor

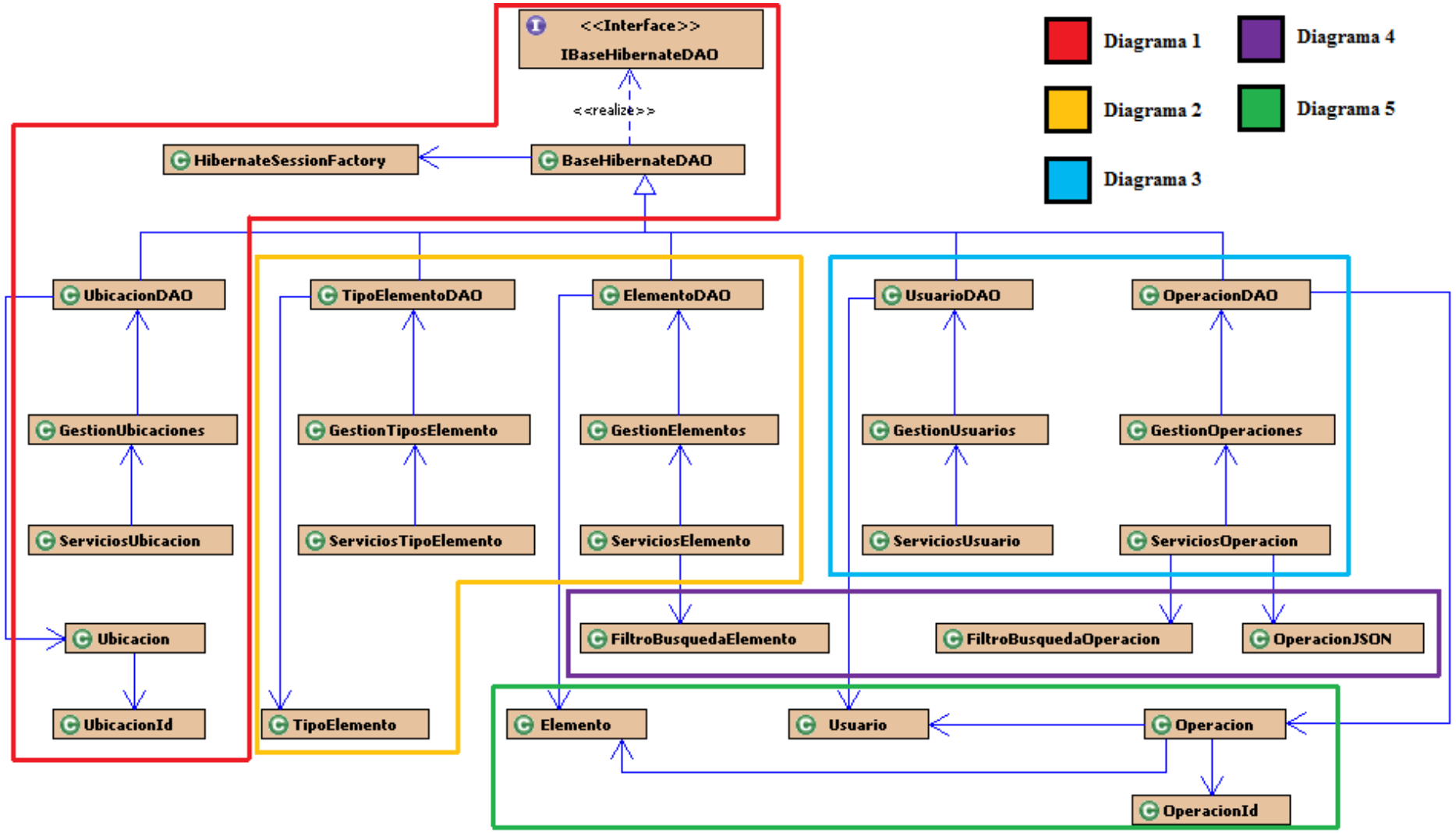


Ilustración 24: Diagrama de clases de la aplicación servidor dividido por zonas

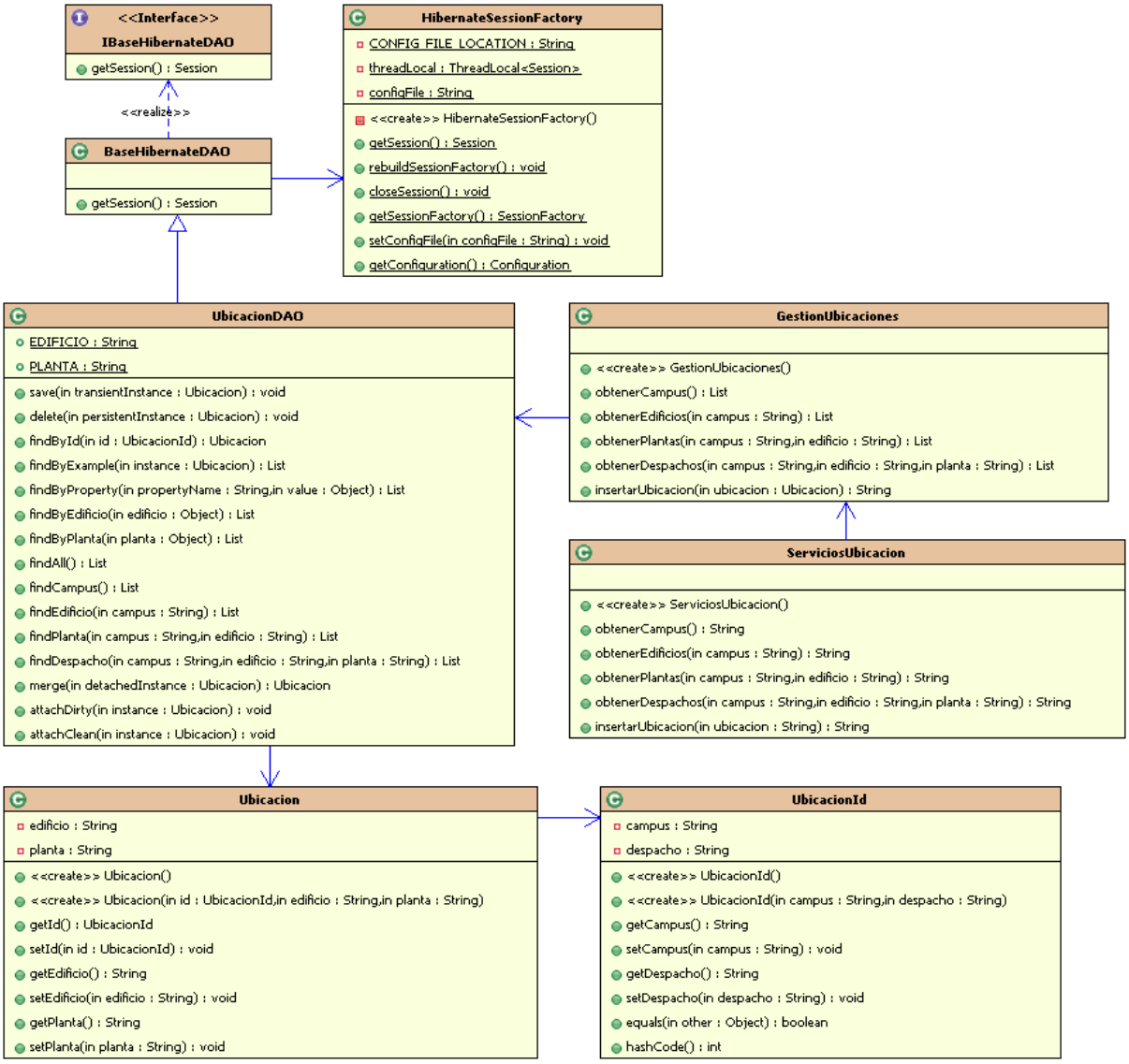


Ilustración 25: Subdiagrama de clases 1 de la aplicación servidor

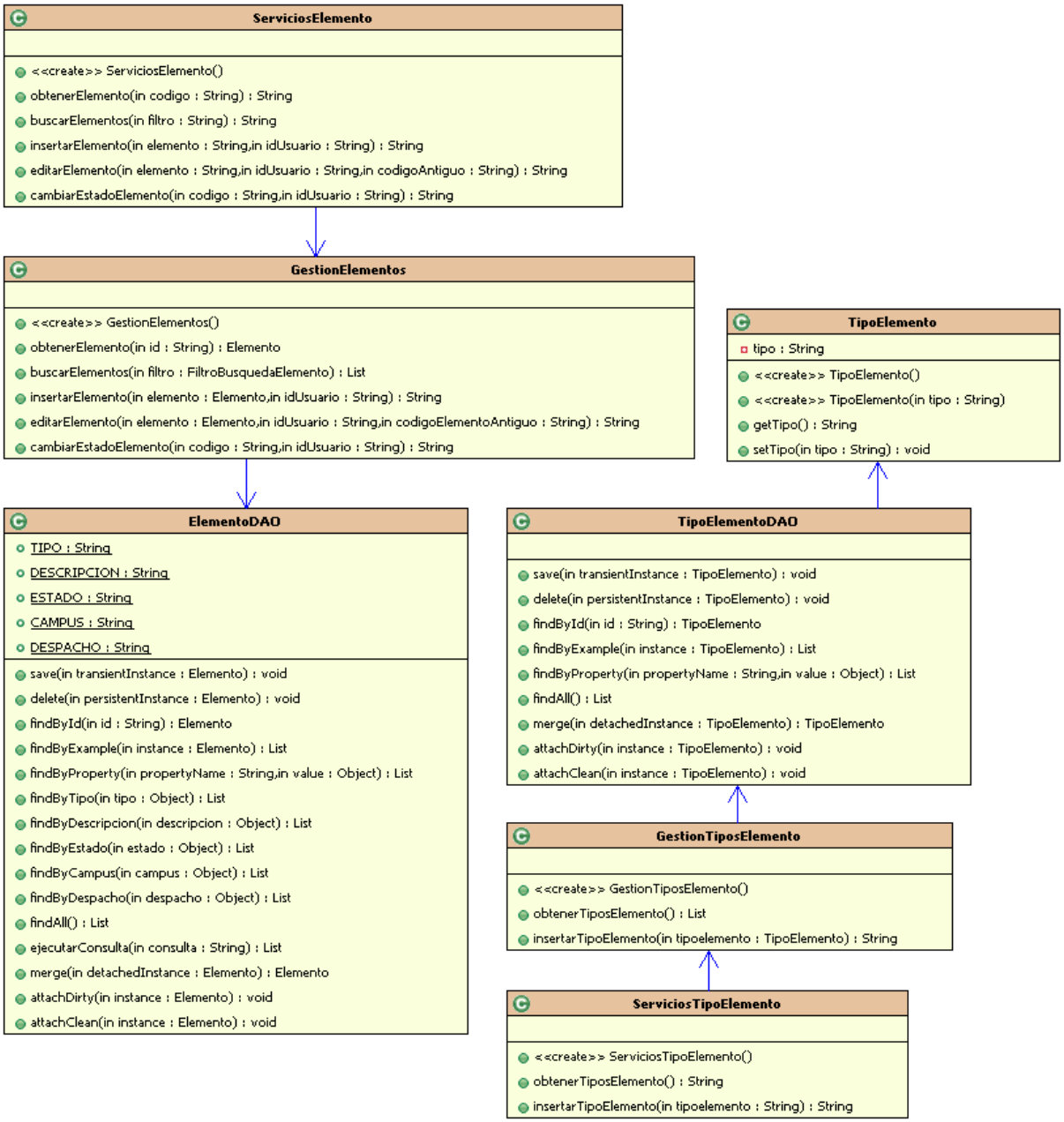


Ilustración 26: Subdiagrama de clases 2 de la aplicación servidor

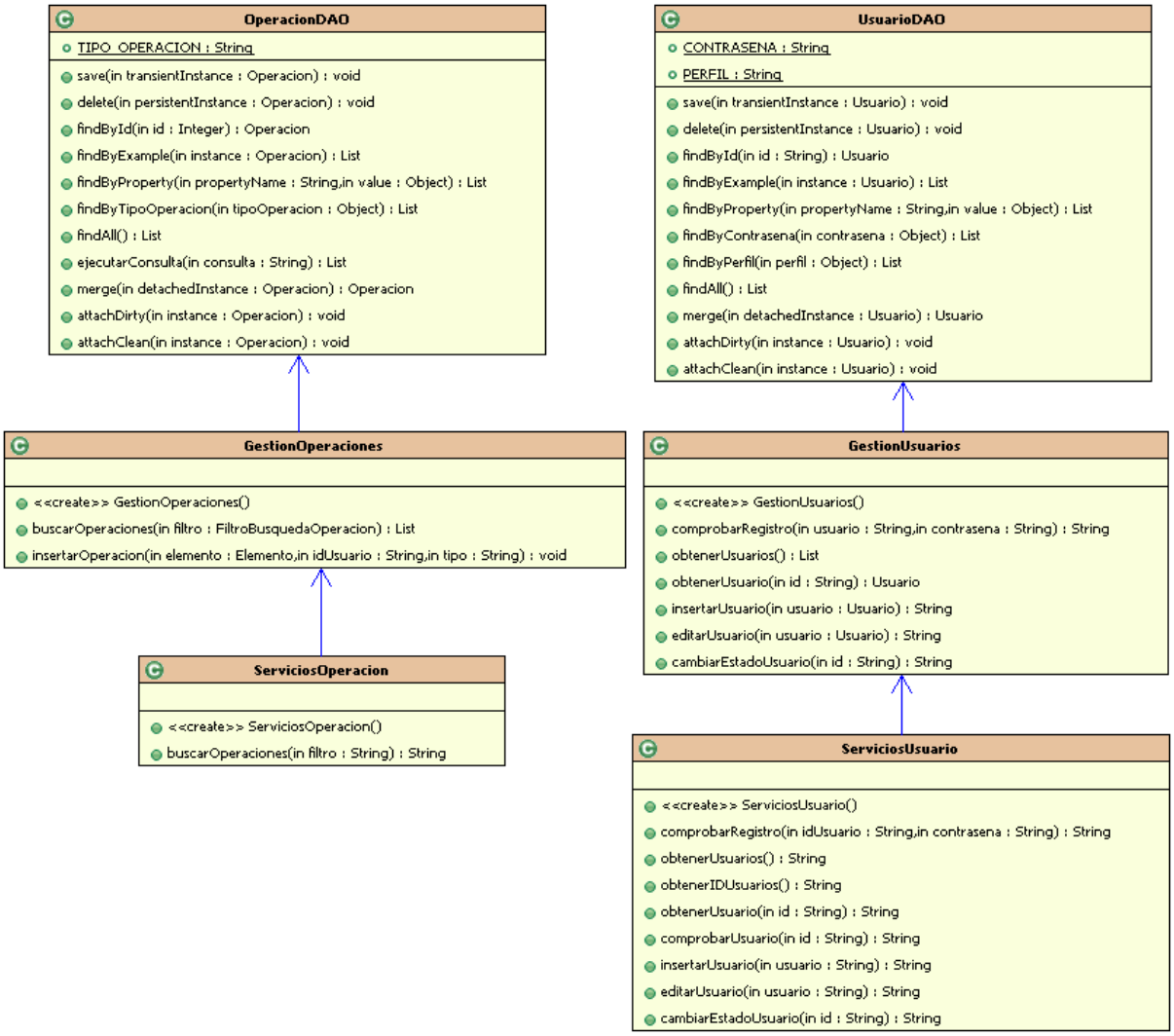


Ilustración 27: Subdiagrama de clases 3 de la aplicación servidor

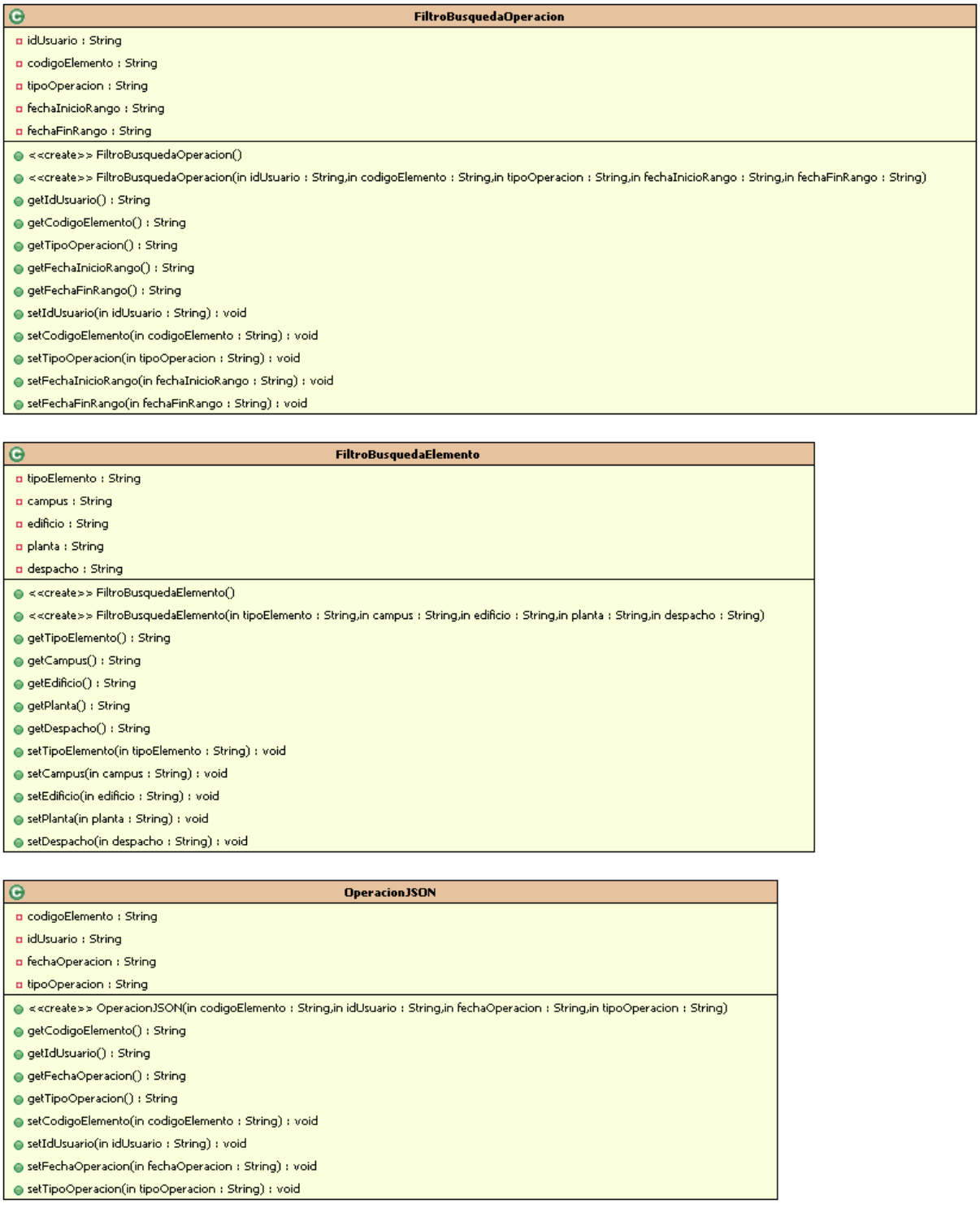


Ilustración 28: Subdiagrama de clases 4 de la aplicación servidor

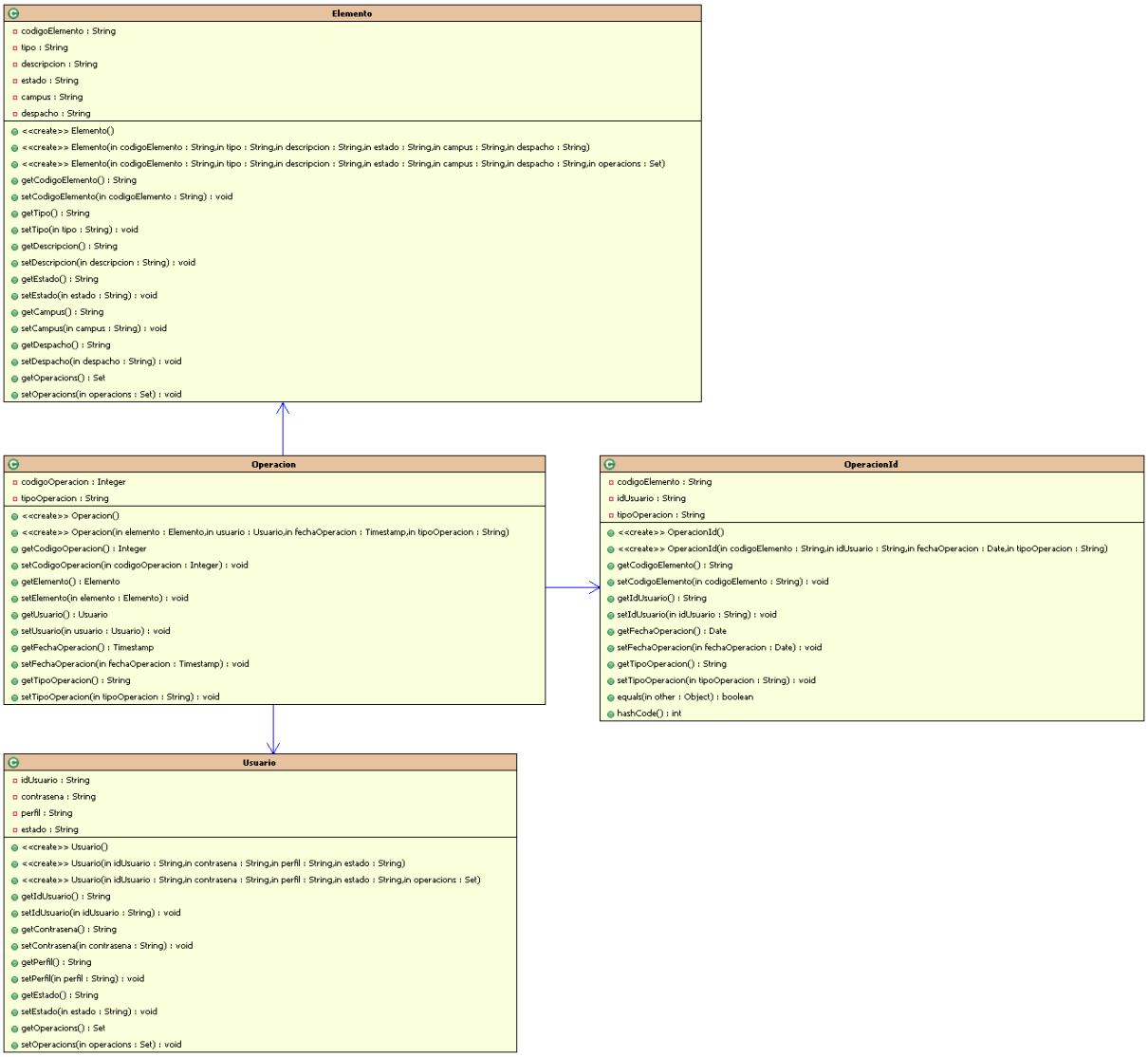


Ilustración 29: Subdiagrama de clases 5 de la aplicación servidor

4.3 DISEÑO DE BASE DE DATOS

En esta sección se mostrará el diagrama relacional de la base de datos implementada para la aplicación servidor. Tras el diagrama, se comentarán algunas consideraciones al respecto de la base de datos del sistema.

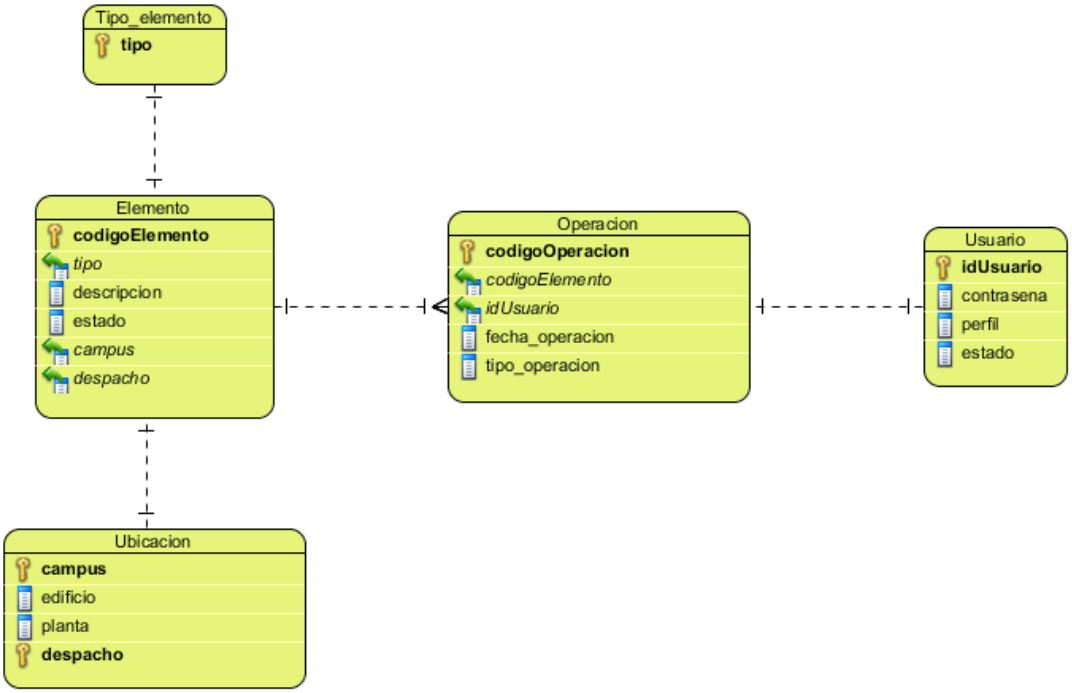


Ilustración 30: Diagrama de BBDD del sistema

Como se puede observar, existe una tabla por cada tipo de datos manejado en el sistema. Las principales son *Elemento* y *Usuario*, tipos de dato gestionados de manera principal en la aplicación. También tenemos la tabla *Operación*, creada para abstraer de una manera más eficaz la información acerca de la interacción de los usuarios con el sistema, ya que dicha información se podría haber introducido en *Elemento*. Por último, tenemos las tablas *Tipo_elemento* y *Ubicacion*, utilizadas para almacenar la información sobre tipos y ubicaciones que se recuperará desde la aplicación cliente para que el usuario tenga que introducir la mínima información al realizar inserciones y consultas de elementos.

Consideraciones generales sobre la BBDD:

- El campo *estado* de la tabla *Elemento* es un enumerado que tiene los valores 'Activo' o 'Borrado'.
- El campo *contrasena* de la tabla *Usuario* es un hash de 32 bytes.
- El campo *perfil* de la tabla *Usuario* es un enumerado que tiene los valores 'Usuario' o 'Administrador'.
- El campo *estado* de la tabla *Usuario* es un enumerado que tiene los valores 'Activo' o 'Inactivo'.
- El campo *codigoOperacion* de la tabla *Operacion* es un campo generado automáticamente mediante autoincremento.
- El campo *fecha_operacion* de la tabla *Operacion* es de tipo *Timestamp*, para poder guardar fecha y hora.
- El campo *tipo_operacion* de la tabla *Operacion* es un enumerado que tiene los valores 'Creacion', 'Edicion', 'Borrado' o 'Restauracion'.
- La clave primaria de la tabla *Ubicación* está formada por los campos *campus* y *despacho*. Se ha considerado así ya que en distintos campus puede haber dos despachos con la misma nomenclatura, por lo que la clave no podía ser solo el despacho.

4.4 DECISIONES DE DISEÑO Y DETALLES DE IMPLEMENTACIÓN

En esta sección se van a mostrar algunos detalles de implementación de aspectos complejos de la aplicación que requirieron alguna investigación adicional para su resolución. Además, se van a comentar algunas decisiones de diseño tomadas que influyen en algunos aspectos del funcionamiento de la aplicación.

Detalles de implementación

1) Como se ha explicado anteriormente, la aplicación pretende que el usuario minimice la inserción de información ya existente en el servidor. Esto, unido a las funcionalidades propias de la aplicación, provoca frecuentes peticiones al servidor mediante llamadas a los servicios Web. Para evitar que la aplicación quede congelada durante la ejecución de estas llamadas, se decidió realizar las peticiones a los servicios Web en un hilo en segundo plano, y mostrar un diálogo de carga en el hilo principal. Aquí surgió el problema.

   En *Java*, cuando se necesita crear hilos extras, es habitual crear clases que extiendan de *Thread* o implementen la interfaz *Runnable*. En *Android*, el uso de hilos tiene un inconveniente, y es que sólo se pueden cambiar las propiedades de un componente gráfico de la interfaz en el proceso que los ha creado, esto es, el hilo principal. Si tratamos de cambiar, empleando otro hilo, el texto de un *TextView*, por ejemplo, nos encontraremos con una excepción. Esto supone un problema que impide la ejecución de las peticiones a servicio Web en un hilo en segundo plano, ya que habitualmente se necesita acceder a elementos de la interfaz.

   Tras investigar, se ha encontrado una solución válida y elegante que consiste en emplear la clase *AsyncTask*[16]. Esta clase está diseñada con el objetivo de realizar tareas asíncronas y cuenta con un método, *onPostExecute()*, que es llamado cuando se termina la tarea y que permite llevar a cabo cambios sobre los elementos de la interfaz. Las clases que se describieron en el punto anterior y que modelan las tareas realizadas por la aplicación a través de servicio Web heredan todas de *AsyncTask*.

   A continuación se puede ver el prototipo de clases que heredan de *AsyncTask*.

```
public class MiTarea extends AsyncTask<Params, Progress, Result> {

    protected Result doInBackground(Params… p) {
    }

    protected void onPreExecute() {
    }

    protected void onProgressUpdate(Progress… values) {
    }

    protected void onPostExecute(Result result) {
    }

}
```

2) En el funcionamiento normal de la aplicación, cuando se navega entre pantallas, es normal pasar datos entre distintas actividades, dado que cada actividad controla una pantalla de la interfaz gráfica. La técnica habitual de paso de datos entre actividades es cargar un *bundle* en el *intent* con el que se llama a la actividad, para recuperarlo después en el *onCreate()* de la actividad llamada. Cuando los datos que se pasan son tipos básicos no hay ningún problema; el problema surge cuando lo que se pasan son objetos personalizados, como es el caso de la aplicación con elementos o usuarios, por ejemplo.

   La solución a este problema es usar la clase *Parcelable*. Para ello, las clases que modelan los tipos de datos personalizados deben implementar esta interfaz, y sobrescribir el método *writeToParcel()* e implementar un atributo estático llamado *CREATOR*. De esta manera, *Android* tiene una noción de la estructura interna de los objetos personalizados, por lo que al recuperar el *bundle* en la clase llamada, sabe cómo reconstruir los objetos, y éstos se pueden utilizar de la manera habitual sin ningún problema.

   A continuación se puede ver el prototipo de clases que implementan *Parcelable*.

```
public class ParcelData implements Parcelable {

   public static final Parcelable.Creator CREATOR = new
                                          Parcelable.Creator() {

           public ParcelData createFromParcel(Parcel in) {
                        return new ParcelTest(in);
           }

           public ParcelData[] newArray(int size) {
                        return new ParcelTest[size];
           }
    };


    public void writeToParcel(Parcel dest, int flags) {
    }

}
```

3) Como se ha podido observar en puntos anteriores, el sistema utiliza fechas tipo *timestamp* para guardar la fecha y hora de las operaciones realizadas en el sistema, y también un rango de fechas al realizar la consulta de operaciones. Por lo tanto, es necesario que los *timestamp* viajen en las peticiones a los servicios Web. El problema surgió porque la herramienta utilizada para serializar y deserializar *JSON*, *Gson*, no es capaz de tratar fechas, y provoca errores y pérdida de información en las comunicaciones.

   La solución adoptada fue no enviar datos tipo fecha, sino strings que contengan la información de los *timestamp*. El proceso de conversión de *timestamp* a *string* y viceversa es muy sencillo utilizando una clase llamada *SimpleDateFormat*[17], que permite introducir patrones de representación de fechas, lo que también es útil para imprimir las fechas en la interfaz gráfica en un formato legible para el usuario, y con el método *parse()* se puede realizar la transformación sin pérdida de información alguna. Así pues, el tipo *timestamp* solo es utilizado en el servidor, a la hora de guardar en la base de datos y para realizar las consultas solicitadas.

Decisiones de diseño

- Cuando se modifica el código de un elemento (cambio de etiqueta), el elemento con el código antiguo se marca como borrado y se añade un nuevo elemento con el nuevo código.

- No se permite la edición de ubicaciones existentes en el sistema.

- No se permite la edición de tipos de elemento existentes en el sistema.

- No se permite modificar el identificador de un usuario una vez dado de alta.

- No se guarda ningún dato personal de un usuario (nombre y apellidos, por ejemplo), ya que el trabajo de inventario está bastante orientado a cuentas de usuario utilizadas por varias personas (becarios que van cambiando).

- Se ha decidido integrar la aplicación *BarcodeScanner* para leer códigos debido a su gratuidad en el *Android Market* y a que se proporcionan unas clases de integración en aplicaciones, lo que suponía mucho menos trabajo que desarrollar un lector de código de barras embebido en la aplicación.

5. PRUEBAS

5.1 PRUEBAS DE ACEPTACIÓN

En esta sección se van a presentar las pruebas finales hechas a la aplicación y que han servido para verificar la corrección de todas las funcionalidades del sistema y para validar la lista de requisitos expuestos anteriormente, lo que se conoce como pruebas de aceptación.

Estas pruebas se van a presentar en forma de tabla. Los campos de la tabla tienen el siguiente significado:

- Identificador: será un código del tipo *PA-XY*, siendo *XY* un número comenzando en "01", hasta el que sea necesario para completar todas las pruebas.

- Descripción: descripción completa de la prueba realizada.

- Resultado: descripción del resultado de la prueba. Puede referirse a los cambios observados en el sistema con posterioridad.

El entorno de pruebas ha sido un HTC Desire con Android 2.2 Froyo.

NOTA: no se tendrá en cuenta para la realización de las pruebas los posibles errores provenientes del servidor, como fallos de conexión, de base de datos, etc.

Identificador	PA-01
Descripción	Se va a comprobar que el proceso de *login* de la aplicación es correcto. Los procesos realizados han sido los siguientes: 1) Se introducen los datos correctos de un perfil usuario. 2) Se introducen los datos correctos de un perfil administrador. 3) Se introduce un identificador de usuario inexistente. 4) Se introduce un identificador de usuario correcto pero con contraseña incorrecta.

	5) Se introducen los datos correctos de un usuario dado de baja.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el menú principal del perfil usuario. 2) Se muestra el menú principal del perfil administrador. 3) Sobre la pantalla principal de la aplicación, se muestra el mensaje "El usuario no existe". 4) Sobre la pantalla principal de la aplicación, se muestra el mensaje "La contraseña no es correcta". 5) Sobre la pantalla principal de la aplicación, se muestra el mensaje "El usuario introducido se encuentra inactivo".

Tabla 52: Prueba de aceptación PA-01

Identificador	PA-02
Descripción	Se va a comprobar que el proceso añadir un nuevo elemento funciona de manera correcta. Los procesos realizados han sido los siguientes: 1) Se introducen correctamente todos los datos en el formulario y se confirma la inserción. 2) Se introduce un código de elemento erróneo y se trata de avanzar hasta la ubicación. 3) No se selecciona ningún tipo de elemento y se trata de avanzar hasta la ubicación. 4) No se introduce ninguna descripción y se trata de avanzar hasta la ubicación. 5) No se introduce la ubicación completa y se confirma la inserción. 6) Se intenta salir al menú principal habiendo introducido alguna información del nuevo elemento, pero sin insertarlo.
Resultado	Los resultados de los distintos procesos han sido los esperados:

	1) Se muestra el menú principal del perfil usuario, y el mensaje "Se ha insertado correctamente el elemento". 2) Sobre la pantalla de inserción de nuevo elemento, se muestra el mensaje "El código introducido es erróneo". 3) Sobre la pantalla de inserción de nuevo elemento, se muestra el mensaje "No se ha seleccionado ningún tipo de elemento". 4) Sobre la pantalla de inserción de nuevo elemento, se muestra el mensaje "No se ha introducido ninguna descripción". 5) Sobre la pantalla de inserción de ubicación de nuevo elemento, se muestra el mensaje "No se ha seleccionado la ubicación completa". 6) Sobre la pantalla en la que se esté en ese momento (inserción de información de nuevo elemento o inserción de ubicación de nuevo elemento), se muestra un diálogo informando de la posible pérdida de información si se sale, y dos botones, uno para confirmar la salida y otro para cancelar la acción. Ambos botones funcionan correctamente.

Tabla 53: Prueba de aceptación PA-02

Identificador	PA-03
Descripción	Se va a comprobar que el proceso de consulta de inventario de la aplicación es correcto. Los procesos realizados han sido los siguientes: 1) Se ejecuta una consulta sin introducir ningún dato. 2) Se ejecuta una consulta escribiendo un código de elemento correcto. 3) Se ejecuta la lectura del código de barras del elemento. 4) Se ejecuta una consulta escribiendo un código de elemento incorrecto. 5) Se ejecuta una consulta escribiendo un código de elemento correcto y

	especificando algún otro filtro. 6) Se ejecuta una consulta especificando solo filtros (uno o los dos). NOTA: Este proceso en el perfil administrador es análogo, y aunque se ha probado también su correcto funcionamiento, no se va a mostrar otra tabla con esa prueba.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el mensaje "No se ha introducido código de elemento ni ningún filtro". 2) Se muestra la vista en detalle del elemento recuperado, si existe, y si no el mensaje "No existe ningún elemento con ese código". 3) Tras completar la lectura del código, se muestra la vista en detalle del elemento recuperado, si existe, y si no el mensaje "No existe ningún elemento con ese código". 4) Se muestra el mensaje "El código introducido no es válido". 5) Se muestra un diálogo informando de que la información de los filtros no se tendrá en cuenta, y tras pulsar en el botón, se muestra una lista si hay más de un resultado, la vista en detalle si solo hay un resultado, o el mensaje "La búsqueda no produjo ningún resultado" si no hay resultados. 6) Se muestra una lista si hay más de un resultado, la vista en detalle si solo hay un resultado, o el mensaje "La búsqueda no produjo ningún resultado" si no hay resultados.

Tabla 54: Prueba de aceptación PA-03

Identificador	PA-04
Descripción	Se va a comprobar que el proceso de editar elemento es correcto. Los procesos realizados han sido los siguientes: 1) Se cambia el código de elemento y se

	introduce un código correcto. 2) Se cambia el código de elemento y se introduce un código incorrecto. 3) Se cambia tipo, descripción o ubicación de manera correcta. 4) Se cambia tipo, descripción o ubicación de manera incorrecta. 5) Se intenta salir al menú principal después de cambiar alguna información del elemento, y antes de confirmar el cambio. 6) Se intenta actualizar el elemento sin haber cambiado ninguna información. NOTA: Este proceso en el perfil administrador es análogo, y aunque se ha probado también su correcto funcionamiento, no se va a mostrar otra tabla con esa prueba.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el mensaje "Se ha actualizado el elemento correctamente". El antiguo elemento se conserva en la BBDD, con el estado "Borrado" y se añade un nuevo elemento con el nuevo código, que es el que se muestra en pantalla. 2) Se muestra el mensaje "El código introducido no es válido". 3) Se muestra el mensaje "Se ha actualizado el elemento correctamente". 4) Se muestra un mensaje dependiendo del error concreto, no se selecciona tipo, no se introduce descripción, o no se selecciona la ubicación completa. 5) Se muestra un diálogo informando de que se pueden perder los cambios si se sale, y dos botones, uno para confirmar la salida y otro para cancelar. Ambos botones funcionan correctamente. Si se sale no se actualizará el elemento. 6) Se muestra el mensaje "No es necesario guardar. No ha habido cambios".

Tabla 55: Prueba de aceptación PA-04

Identificador	PA-05
Descripción	Se va a comprobar que el proceso de borrar/restaurar elemento es correcto. Los procesos realizados han sido los siguientes: 1) Cuando el estado del elemento es "Activo", se pulsa el botón de "Borrar elemento". 2) Cuando el estado del elemento es "Borrado", se pulsa el botón de "Restaurar elemento". NOTA: Este proceso en el perfil administrador es análogo, y aunque se ha probado también su correcto funcionamiento, no se va a mostrar otra tabla con esa prueba.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el mensaje "Se ha borrado correctamente el elemento". Se cambia en la interfaz el estado del elemento y el texto del botón. 2) Se muestra el mensaje "Se ha restaurado correctamente el elemento". Se cambia en la interfaz el estado del elemento y el texto del botón.

Tabla 56: Prueba de aceptación PA-05

Identificador	PA-06
Descripción	Se va a comprobar que el proceso de consulta de operaciones de la aplicación es correcto. Los procesos realizados han sido los siguientes: 1) Se ejecuta una consulta sin especificar ningún filtro. 2) Se ejecuta una consulta especificando uno o más filtros de manera correcta (el filtro de usuario solo lo podrá utilizar el perfil administrador, el perfil usuario verá solo sus operaciones). 3) Se especifica una fecha de fin de rango menor a la fecha de inicio. NOTA: Este proceso en el perfil administrador es análogo, y aunque se ha probado también su correcto

	funcionamiento, no se va a mostrar otra tabla con esa prueba.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el mensaje "No se ha especificado ningún filtro". 2) Se muestra una lista con los resultados obtenidos, o el mensaje "La búsqueda no produjo resultados" si no hay resultados. 3) Se muestra el mensaje "La fecha de fin de rango debe ser igual o mayor a la fecha de inicio".

Tabla 57: Prueba de aceptación PA-06

Identificador	PA-07
Descripción	Se va a comprobar que el proceso añadir un nuevo usuario funciona de manera correcta. Los procesos realizados han sido los siguientes: 1) Se introducen correctamente todos los datos en el formulario y se confirma la inserción. 2) Se confirma la inserción del usuario sin introducir ningún identificador. 3) Se introduce un identificador de usuario ya en uso. 4) Se deja sin completar alguna de las casillas de contraseña, o las dos. 5) El contenido de las dos casillas de contraseña no coincide. 6) No se selecciona ningún perfil. 7) Se intenta salir al menú principal habiendo introducido alguna información del nuevo usuario, pero sin insertarlo.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el menú principal del perfil administrador, y el mensaje "Se ha insertado correctamente el usuario". 2) Sobre la pantalla de inserción de nuevo usuario, se muestra el mensaje "No se ha introducido un identificador".

	3) Sobre la pantalla de inserción de nuevo usuario, se muestra el mensaje "El identificador introducido ya está en uso". 4) Sobre la pantalla de inserción de nuevo usuario, se muestra el mensaje "No se ha completado alguna de las casillas de contraseña". 5) Sobre la pantalla de inserción de nuevo usuario, se muestra el mensaje "El contenido de las dos casillas de contraseña no coincide". 6) Sobre la pantalla de inserción de nuevo usuario, se muestra el mensaje "No se ha seleccionado el perfil del usuario". 7) Se muestra un diálogo informando de que se perderá la información del nuevo usuario si se sale, y dos botones, uno para confirmar la salida y otro para cancelar. Los dos botones funcionan correctamente.

Tabla 58: Prueba de aceptación PA-07

Identificador	PA-08
Descripción	Se va a comprobar que el proceso de editar usuario es correcto. Los procesos realizados han sido los siguientes: 1) Se cambia la contraseña del usuario correctamente. 2) Se cambia la contraseña del usuario, pero se deja alguna de las casillas, o las dos, sin rellenar. 3) Se cambia la contraseña del usuario, pero el contenido de las dos casillas no coincide. 4) Se cambia el perfil del usuario correctamente. 5) Se cambia el perfil del usuario, pero se deja sin seleccionar ninguno. 6) Se intenta salir al menú principal después de cambiar alguna información del usuario, y antes de confirmar el cambio. 7) Se intenta actualizar el usuario sin haber cambiado ninguna información.

Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el mensaje "Se ha actualizado el usuario correctamente". 2) Se muestra el mensaje "No se ha rellenado alguna casilla de contraseña". 3) Se muestra el mensaje "El contenido de las casillas de contraseña no coincide". 4) Se muestra el mensaje "Se ha actualizado el usuario correctamente". 5) Se muestra el mensaje "No se ha seleccionado ningún perfil". 6) Se muestra un diálogo informando de que se pueden perder los cambios si se sale, y dos botones, uno para confirmar la salida y otro para cancelar. Ambos botones funcionan correctamente. Si se sale no se actualizará el usuario. 7) Se muestra el mensaje "No es necesario guardar. No ha habido cambios".

Tabla 59: Prueba de aceptación PA-08

Identificador	PA-09
Descripción	Se va a comprobar que el proceso de activar/desactivar usuario es correcto. Los procesos realizados han sido los siguientes: 1) Cuando el estado del usuario es "Activo", se pulsa el botón de "Desactivar usuario". 2) Cuando el estado del usuario es "Inactivo", se pulsa el botón de "Activar usuario".
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el mensaje "Se ha desactivado correctamente el usuario". Se cambia en la interfaz el estado del usuario y el texto del botón. 2) Se muestra el mensaje "Se ha activado correctamente el usuario". Se cambia en la interfaz el estado del usuario y el texto del botón.

Tabla 60: Prueba de aceptación PA-09

Identificador	PA-10
Descripción	Se va a comprobar que el proceso de añadir ubicación es correcto. Los procesos realizados han sido los siguientes: 1) Se introduce la ubicación correctamente y se confirma la inserción. 2) Se trata de confirmar la inserción sin completar la introducción de datos de la nueva ubicación. 3) Se trata de salir al menú principal después de introducir información sobre la nueva ubicación, sin guardarla.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se muestra el menú de gestión de recursos y el mensaje "Se ha insertado correctamente la ubicación". 2) Se muestra el mensaje "No se ha insertado la ubicación correctamente". 3) Se muestra un diálogo informando de que se puede perder la información si se sale, y dos botones, uno para confirmar la salida y otro para cancelar. Ambos botones funcionan correctamente.

Tabla 61: Prueba de aceptación PA-10

Identificador	PA-11
Descripción	Se va a comprobar que el proceso de añadir tipo de elemento es correcto. Los procesos realizados han sido los siguientes: 1) Se introduce el nuevo tipo de elemento correctamente. 2) Se introduce un tipo de elemento ya existente.
Resultado	Los resultados de los distintos procesos han sido los esperados: 1) Se actualiza la lista de tipos de elemento para añadir el nuevo, y se muestra el

	mensaje “Se ha insertado el nuevo tipo de elemento correctamente”. 2) Se muestra el mensaje “El tipo de elemento introducido ya existe”.

Tabla 62: Prueba de aceptación PA-11

5.2 MATRIZ DE TRAZABILIDAD REQUISITOS – PRUEBAS

RU / PA	RF-01	RF-02	RF-03	RF-04	RF-05	RF-06	RF-07	RF-08	RF-09	RF-10	RF-11	RF-12	RF-13
PA-01	X												
PA -02		X											
PA -03			X										
PA -04				X									
PA -05					X	X							
PA -06							X						
PA -07								X					
PA -08									X				
PA -09										X	X		
PA -10													X
PA -11												X	

Tabla 63: Matriz de trazabilidad requisitos-pruebas

6. CONCLUSIONES Y LÍNEAS FUTURAS

En este apartado se presentarán las conclusiones posteriores a la realización del proyecto y se revisarán los objetivos planteados en el punto de introducción, para comprobar el grado de satisfacción de cada uno de ellos. Asimismo, se presentan futuras líneas de trabajo para posibles ampliaciones posteriores del proyecto.

6.1 CONCLUSIONES

Como se ha comentado, en el primer capítulo de este documento, se presentaron una serie de objetivos que había que cumplir para perseguir un objetivo principal, la realización de una aplicación de gestión de inventario. Así pues, a continuación se exponen las conclusiones respecto al cumplimiento de dichos objetivos:

- Se han identificado exitosamente los procesos principales que componen la gestión de inventario de cualquier empresa.

- Se ha conseguido almacenar en una base de datos toda la información necesaria para ejecutar el sistema con total normalidad y eficacia.

- Se ha diseñado e implementado de manera satisfactoria una base de datos que actúe de ejemplo en el sistema desarrollado, teniendo en cuenta la información identificada anteriormente.

- Se ha construido exitosamente una aplicación servidor que cumple con su cometido de manera eficiente, accediendo a la base de datos implementada, y comunicándose correctamente con los dispositivos móviles.

- Se ha conseguido el conocimiento necesario acerca de la arquitectura de *Android* para poder implementar una aplicación adecuada a los requerimientos.

- Se ha diseñado e implementado una aplicación *Android* para ejecutar en los dispositivos móviles que cumple con todas las especificaciones necesarias para su correcto funcionamiento, implementando todos los procesos de gestión de inventario, comunicándose con el servidor de manera correcta, y presentando al usuario una interfaz sencilla e

intuitiva, con la que es fácil cumplir los cometidos para los que se ha desarrollado.

- Se ha realizado una batería de pruebas completa y exhaustiva, que ha permitido validar todos los requisitos planteados inicialmente, y que ha permitido concluir que el sistema funciona correctamente en todos los aspectos solicitados.

- Se ha documentado todo el código de las aplicaciones, en formato *Javadoc*, y el proceso de realización del proyecto, en el presente documento, de manera completa y satisfactoria para su futuro uso o ampliación.

Así pues, habiendo cumplido todos los objetivos propuestos, se puede concluir que se ha alcanzado satisfactoriamente el objetivo principal de la realización de una aplicación de gestión de inventario eficaz, eficiente, y sencilla e intuitiva para los usuarios.

Desde un punto de vista personal, el autor de este documento ha podido comprobar con satisfacción cómo su trabajo ha llegado a buen puerto y se han podido superar las dificultades encontradas con trabajo, tesón e investigación. Se ha descubierto, asimismo, que el desarrollo de aplicaciones móviles para *Android* es realmente muy interesante y que tiene un futuro lleno de posibilidades. Por todo ello, a la finalización de este proyecto se tiene una maravillosa sensación de plenitud ante lo que es el último gran paso para la finalización de la carrera, y un gran orgullo por el trabajo bien hecho.

En conclusión, el proyecto ha colmado de manera sobrada todas las expectativas puestas en él desde un comienzo, tanto desde el punto de vista del buen funcionamiento de todo el sistema, hasta las impresiones y satisfacciones que ha dejado todo el proyecto en el plano personal del autor.

6.2 LÍNEAS FUTURAS DE TRABAJO

En este apartado se van a presentar algunas posibles líneas de investigación de cara a posibles futuras ampliaciones de este proyecto:

1) Conexión Socket Seguro SSL: en esta primera versión no se ha considerado como máxima prioridad la seguridad en las conexiones con el servidor, prefiriendo centrar los esfuerzos en otras funcionalidades. De todas formas, es evidente que sería una de las primeras ampliaciones que sería recomendable realizar al sistema.
   Con la disposición actual del sistema, sería realmente sencillo implementar la seguridad en las conexiones: tan solo habría que realizar unos certificados y validarlos en el servidor y en cliente, realizando las llamadas desde el cliente con *https* en lugar de *http*. En la siguiente ilustración se muestra el proceso de manera gráfica:

Ilustración 31: Establecimiento de una conexión SSL segura

2) Ejecución en otras plataformas: se ha decidido utilizar *Android* para la realización del proyecto por su amplia utilización hoy en día, pero podría ser interesante en un futuro ampliar la aplicación para su ejecución en otros entornos, como *iPhone*, *Windows Phone*, e incluso en otros dispositivos como *Tablets*.

3) Comunicaciones 3G: en la versión actual la comunicación con el servidor solo puede realizarse mediante WiFi, pero una posible ampliación podría dar lugar a permitir comunicaciones por 3G, teniendo en cuenta además que ahora es muy común que las personas u organizaciones que cuentan con dispositivos inteligentes, o *smartphones*, tengan también contratada una tarifa plana de datos.

4) Internacionalización: dado que todas las empresas tienen procesos de gestión de inventario, obviamente incluyendo empresas extranjeras, puede ser interesante en un futuro ampliar la aplicación para que soporte más idiomas, lo que daría repercusión internacional y apertura de nuevos mercados y potenciales usuarios.

5) Visor de facturas: actualmente la aplicación solo maneja información puramente de inventario, es decir, descripciones de elementos y sus respectivas ubicaciones. Sería interesante que en posteriores versiones la aplicación fuera capaz de manejar y visualizar información económica de los objetos de inventario, como facturas, ya que puede ser útil disponer de dicha información en los lugares en los que se hagan actuaciones de inventario. De realizar esta ampliación, sería imprescindible implementar conexiones seguras, ya que la información económica es muy sensible en las empresas.

6) Módulo de voz: hoy en día, cada vez más se están lanzando aplicaciones que se pueden manejar mediante órdenes de voz. Esta característica sería perfectamente aplicable a este sistema, e incluso en ciertos entornos puede ser más que recomendable, como en grandes almacenes logísticos donde los operarios están continuamente manejando productos y no pueden perder demasiado tiempo actuando físicamente con el dispositivo.

7. PLANIFICACIÓN Y PRESUPUESTO

7.1 PLANIFICACIÓN

En esta sección se detallará la planificación seguida para la elaboración de este proyecto.

En la siguiente tabla se pueden ver las actividades realizadas durante la elaboración del proyecto indicándose la fecha de inicio y finalización de cada una de ellas así como la duración en días de cada actividad.

Tarea	Fecha de inicio	Duración	Fecha de fin
Toma de requisitos	07/03/2011	6 días	14/03/2011
Elección de tecnologías	15/03/2011	5 días	21/03/2011
Diseño de BBDD	22/03/2011	8 días	31/03/2011
Implementación del servidor	01/04/2011	37 días	23/05/2011
Implementación de la aplicación Android	24/05/2011	73 días	01/09/2011
Pruebas y correcciones	02/09/2011	12 días	19/09/2011
Documentación del código	01/04/2011	122 días	19/09/2011
Realización de la memoria	20/09/2011	21 días	18/10/2011
Revisión	19/10/2011	4 días	24/10/2011

Tabla 64: Planificación de las tareas del proyecto

A continuación se muestra la planificación en formato Diagrama de Gantt:

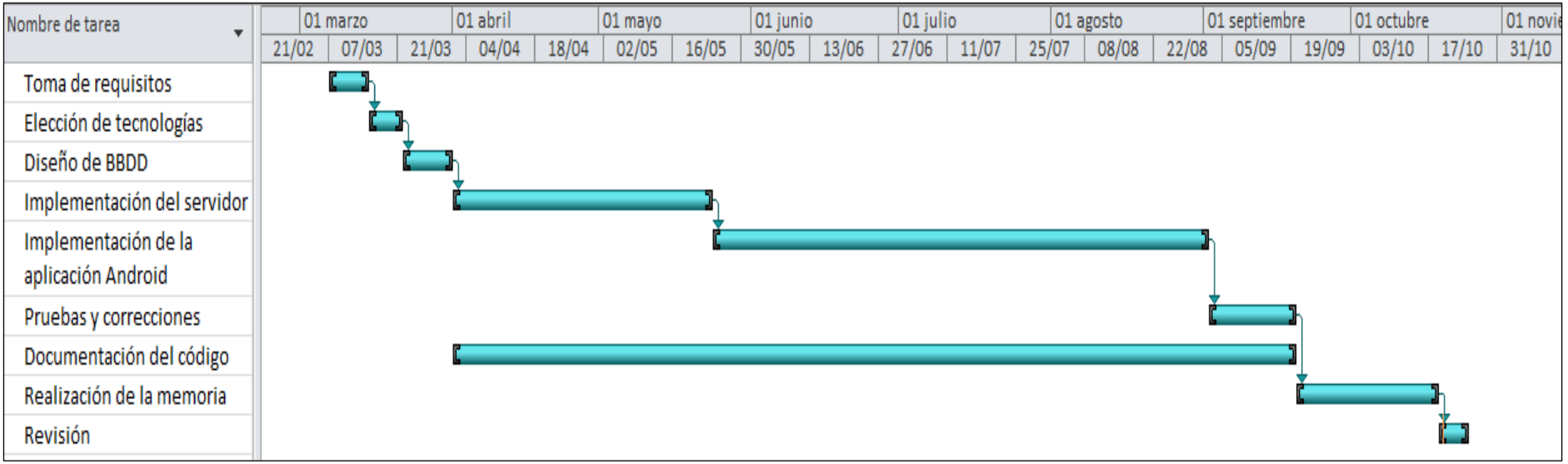


Ilustración 32: Diagrama de Gantt de la planificación

7.2 PRESUPUESTO

En esta sección se va a detallar el presupuesto del proyecto. Para los cálculos de los costes se han tenido en cuenta las siguientes consideraciones:

- La fecha de inicio de proyecto es el 7 de marzo de 2011, y la fecha de fin el 24 de octubre de 2011.
- Se toman como referencia la duración en días de las distintas tareas expuestas en el apartado de planificación. En dicha duración ya se tiene en cuenta que los fines de semana no se trabaja. No se elimina ningún día más del calendario de trabajo por otras razones, como festividades, enfermedad, etc.
- Así pues, el total de días dedicados al proyecto asciende a 166 días, ya que la tarea de documentación de código no se computa al compatibilizarse con las tareas de implementación y pruebas y correcciones.
- La jornada diaria de trabajo ha sido variable, pero se estima una media de 5 horas diarias.
- En total, el número de horas trabajadas en el proyecto es de 830.

7.2.1 COSTES DE PERSONAL

En esta sección se va a especificar los costes asignados a personal del proyecto. La única persona involucrada en la realización de este proyecto ha sido el autor de este documento, que será el que asuma todos los roles especificados a continuación para la realización de las diversas tareas:

Tarea	Rol	Coste (€/hora)	Número de horas	Coste total (€)
Toma de requisitos	Analista	50	30	1.500

Elección de tecnologías	Analista	50	25	1.250
Diseño de BBDD	Analista	50	40	2.000
Implementación del servidor	Programador	30	185	5.550
Implementación de la aplicación Android	Programador	30	365	10.950
Pruebas y correcciones	Ingeniero de pruebas	40	60	2.400
Realización de la memoria	Analista	50	105	5.250
Revisión	Analista	50	20	1.000
TOTAL			**830**	**29.900**

Tabla 65: Costes de personal

7.2.2 COSTES DE HARDWARE

En esta sección se mostrarán los gastos del proyecto imputables a elementos hardware imprescindibles para la realización del mismo, como son el ordenador portátil utilizado para el desarrollo y el *smartphone Android* usado principalmente para las pruebas del sistema:

Concepto	Coste (€)	% Uso dedicado al proyecto	Dedicación (meses)	Período de depreciación (meses)	Coste imputable(*) (€)
Sony Vaio VGN-FW11M	850	100	7	60	99,17
HTC Desire	279	100	7	60	32,55
TOTAL					**131,72**

Tabla 66: Costes de hardware

(*) Fórmula de amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto

7.2.3 COSTES DE SOFTWARE

En esta sección se mostrarán los gastos debidos a licencias de software necesario para el desarrollo del proyecto. El proyecto se ha desarrollado sobre Windows 7 con licencia de estudiante. Se ha utilizado además una licencia de MyEclipse, una licencia de Visual Paradigm y el suite Office 2007:

Concepto	Coste (€)
Licencia de Windows 7	0
Licencia Visual Paradigm Community Edition	0
Licencia de Office 2007	109,90
Licencia de MyEclipse 9	46,40 ($63,55)
TOTAL	**156,3**

Tabla 67: Costes de software

7.2.4 COSTE TOTAL

Teniendo en cuenta lo expuesto en los apartados anteriores, el presupuesto total de la aplicación es el siguiente:

Concepto	Coste (€)
Coste de personal	29.900
Coste de hardware	131,72
Coste de software	156,3
Coste sin IVA	**30.188,02**
Beneficio (50%(*))	**15.094,01**
Margen de riesgo (20%(**))	**6.037,6**
IVA (18%(***))	**9.237,53**
COSTE TOTAL	**60.557,16**

Tabla 68: Coste total del proyecto

(*) Se ha considerado el beneficio adecuado, ya que la rama de las tecnologías de la información es muy sensible a los cambios. Calculado sobre el coste sin IVA.

(**) Se ha considerado el margen de riesgo adecuado, ya que se tiene experiencia en el uso de las herramientas con las que se ha implementado el proyecto, de tal forma que los riesgos en el retraso del proyecto se reducen. Sin embargo, es necesario prever problemas como gastos extra de personal, defectos en los equipos utilizados, etc. Calculado sobre el coste sin IVA.

(***) Aplicado sobre la suma del coste sin IVA, más el beneficio, más el margen de riesgo.

El coste total del proyecto es de **60.557,16 €** (sesenta mil quinientos cincuenta y siete euros con dieciséis céntimos).

8. BIBLIOGRAFÍA

[1] Curso Android: (Marzo 2011)
http://www.maestrosdelweb.com/editorial/curso-android/
http://www.sgoliver.net/blog/?p=1313
http://mobile.tutsplus.com/tutorials/android/android-listview/
http://androideity.com/2011/08/27/controles-de-seleccion-en-android-listas/
http://www.nosolounix.com/2011/05/listview-en-android.html

[2] Open Handset Alliance (OHA): (Marzo 2011)
http://www.openhandsetalliance.com/oha_members.html

[3] Java: (Marzo 2011)
http://www.oracle.com/technetwork/java/index.html

[4] Dalvik Virtual Machine: (Marzo 2011)
http://www.dalvikvm.com/

[5] Herramientas de Desarrollo Android (ADT): (Abril 2011)
http://developer.android.com/sdk/eclipse-adt.html

[6] SQLite: (Abril 2011)
http://www.sqlite.org/

[7] Eclipse: (Abril 2011)
http://www.eclipse.org/

[8] Activity: (Mayo 2011)
http://developer.android.com/reference/android/app/Activity.html

[9] Intent: (Mayo 2011)
http://developer.android.com/reference/android/content/Intent.html

[10] JSON (Javascript Object Notation): (Abril 2011)
http://www.json.org/

[11] REST: (Junio 2011)
http://www.hachisvertas.net/blog/01/2008/07/01/web-services-primeros-pasos-con-eclipse-generar-servicio-web

http://www.codeproject.com/KB/android/webservice-from-android.aspx
http://www.android-peru.com/Connecting-RESTful-Web-Services-Android-Spring-Rest
http://netbeans.org/kb/docs/websvc/rest.html
http://www.myeclipseide.com/documentation/quickstarts/webservices_rest/

[12] JSON vs. XML: (Abril 2011)
http://www.readwriteweb.com/hack/2010/11/json-vs-xml.php

[13] GSON: (Mayo 2011)
http://ubuntulife.wordpress.com/2008/10/24/google-gson-una-libreria-java-para-convertir-json-a-objetos-java-y-viceversa/
http://benjii.me/2010/04/deserializing-json-in-android-using-gson/
http://www.josecgomez.com/2010/04/30/android-accessing-restfull-web-services-using-json/

[14] Barcode Scanner: (Abril 2011)
http://stackoverflow.com/questions/2050263/using-zxing-to-create-an-android-barcode-scanning-app

[15] MVC: (Mayo 2011)
http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf

[16] AsyncTask: (Julio 2011)
http://developer.android.com/reference/android/os/AsyncTask.html

[17] Manejo de fechas: (Julio 2011)
http://carloszuluaga.wikidot.com/articulos:manejo-de-fechas-en-java-i
http://download.oracle.com/javase/1,5.0/docs/api/java/text/SimpleDateFormat.html

ANEXO I: MANUAL DE USUARIO

En este anexo se va a mostrar un recorrido por todas las funcionalidades de la aplicación *Android*, así como explicaciones acerca de su uso.

La estructura de este manual será la siguiente: se mostrarán las interfaces que componen la aplicación, y debajo de cada una de ellas se realizarán algunas aclaraciones necesarias para entender su uso y manejo.

Ilustración 33: Pantalla de acceso de la aplicación

La ilustración 33 presenta la pantalla de login de la aplicación, la primera pantalla que se ve al ejecutar la aplicación. Una vez aquí, un usuario puede acceder como perfil usuario o como administrador.

Esta interfaz cuenta con control de errores: se nos informará de errores en el proceso de acceso, como usuarios inexistentes, contraseñas incorrectas, o si intentamos entrar con los datos de un usuario inactivo.

Ilustración 34: Pantalla de menú de usuario

La ilustración 34 muestra la pantalla del menú de usuario, que es el menú principal que se nos mostrará si accedemos con un usuario perfil usuario.

Ilustración 35: Pantallas de inserción de nuevo elemento

Las pantallas de la ilustración 35 son las correspondientes a la inserción de un nuevo elemento, primera opción del menú de usuario. Como se puede ver, en la primera pantalla se introducen código de elemento, que también se puede escanear, tipo y descripción. En la segunda pantalla se selecciona la ubicación del elemento. Cuando se inserte el elemento, volveremos al menú principal de usuario.

Ilustración 36: Pantallas de consulta de inventario y lista de resultados

En la ilustración 36, la primera pantalla es la de consulta del inventario. Se puede escanear un código para consultar, o añadir filtros. En este caso se consultan los elementos tipo "Monitor", para consultar el elemento que insertamos anteriormente. En la segunda pantalla podemos ver la lista de resultados, entre ellos el elemento insertado antes.

Ilustración 37: Pantallas de vista de elemento en detalle y de editar ubicación

A la izquierda de la ilustración 37 vemos la vista en detalle del elemento que añadimos anteriormente. A la derecha vemos como editamos su ubicación.

Ilustración 38: Pantallas de elemento en detalle activo y borrado

A la izquierda de la ilustración 38 podemos observar la vista en detalle del mismo elemento, después de haber cambiado la ubicación. A la derecha vemos la pantalla tras pulsar el botón de borrar elemento.

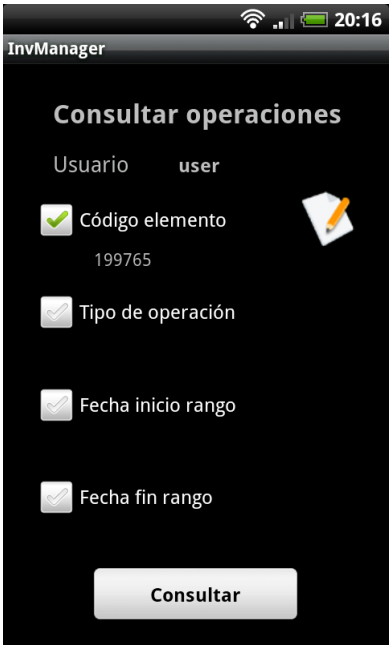


Ilustración 39: Pantalla de consulta de operaciones del perfil usuario

La ilustración 39 presenta la pantalla de consulta de operaciones para perfil usuario. Vemos que se pueden añadir varios filtros de consulta, en este caso queremos ver las operaciones del elemento que añadimos anteriormente.

Ilustración 40: Pantallas de lista de operaciones y vista en detalle

A la izquierda de la ilustración 40 podemos ver la lista de operaciones tras realizar la búsqueda anterior. A esta misma pantalla podemos llegar si pulsamos el botón "Ver operaciones" de la vista en detalle del elemento. En la parte derecha de la ilustración vemos la vista en detalle de la operación. Si pulsamos el botón "Ver elemento" iremos a la pantalla de vista en detalle del elemento.

Ilustración 41: Pantalla de menú de usuario con botón de desconectar y pantalla principal

En la parte izquierda de la ilustración 41 vemos el menú principal de usuario, con la opción de menú de desconectar. En la parte derecha vemos la interfaz principal de la aplicación, que veremos tras desconectar.

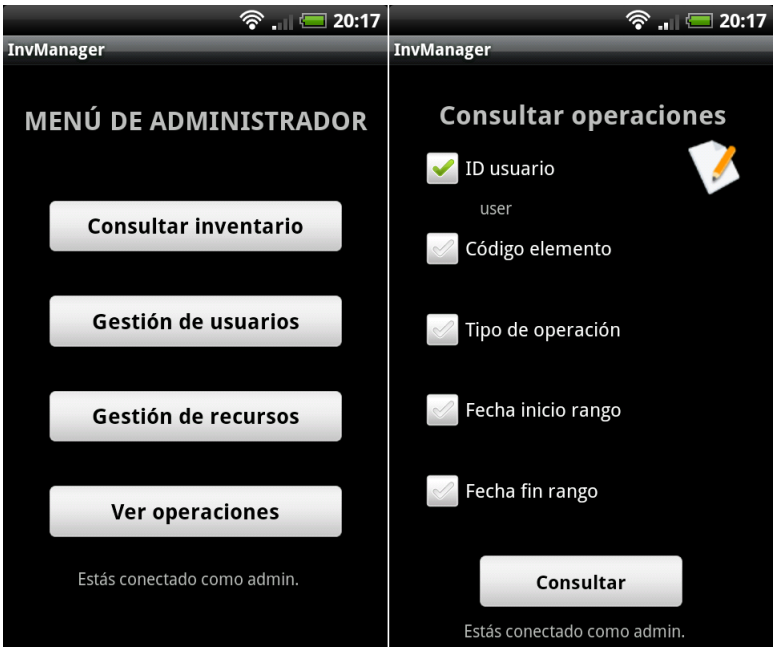


Ilustración 42: Pantallas de menú principal de administrador y consulta de operaciones

En la pantalla de la izquierda de la ilustración 42 vemos el menú principal del perfil administrador. La opción de consultar inventario es igual a la que se vio del perfil usuario. La opción de ver operaciones también es muy similar, pero en la segunda

pantalla podemos observar la diferencia: se puede añadir también un filtro de búsqueda por usuario.

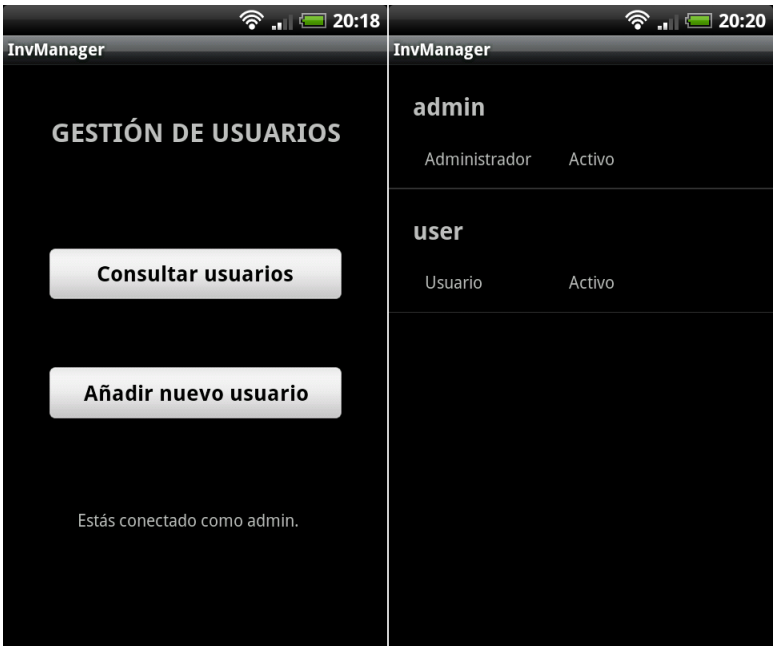


Ilustración 43: Pantallas de gestión de usuarios y lista de usuarios existentes

A la izquierda de la ilustración 43 vemos el menú de opciones de la gestión de usuarios. A la derecha se puede ver la lista de usuarios existentes, que se muestra tras pulsar la opción de consultar usuarios en el menú de gestión de usuarios.

Ilustración 44: Pantallas de vista de usuario en detalle activo e inactivo

En las dos pantallas de la ilustración 44 podemos observar la vista de usuario en detalle, con estado activo e inactivo. El funcionamiento de las ediciones es igual al de los elementos.

Ilustración 45: Pantallas de inserción de nuevo usuario y lista de usuarios existentes

En la pantalla izquierda de la ilustración 45 podemos ver la interfaz para insertar un nuevo usuario. En la derecha podemos ver la lista de usuarios existentes, y cómo el nuevo usuario ha sido añadido.

Ilustración 46: Pantalla de gestión de recursos y lista de tipos de elemento

A la izquierda de la ilustración 46 vemos la pantalla principal de la gestión de recursos. En la derecha se ve la lista de los tipos de elemento existentes, con la opción de añadir uno nuevo.

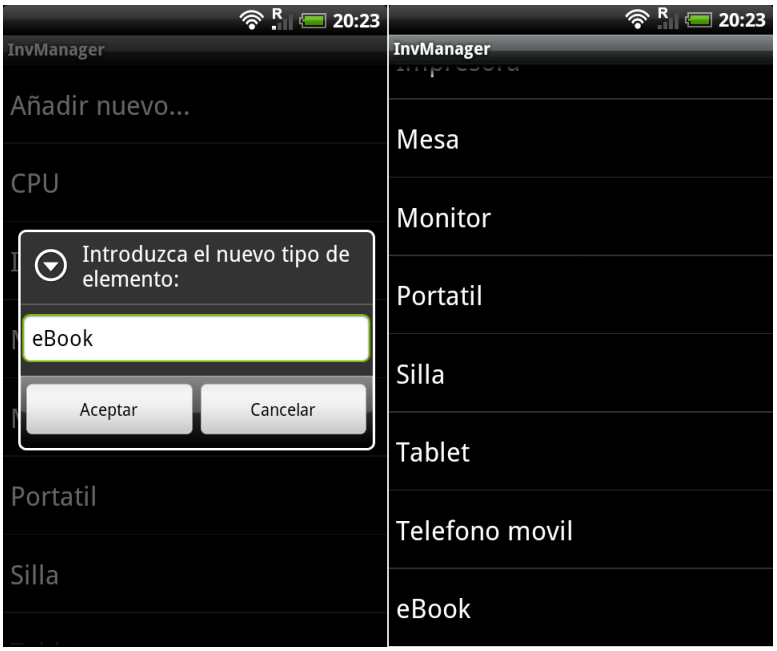


Ilustración 47: Pantallas de añadir nuevo tipo de elemento y lista actualizada

En la pantalla de la izquierda de la ilustración 47 vemos cómo se añade un nuevo tipo de elemento, y en la derecha vemos la lista de tipos de elemento actualizada, incluyendo el que se acaba de añadir.

Ilustración 48: Pantalla de añadir ubicación

La ilustración 48 muestra la pantalla para añadir una nueva ubicación. Cuando se haya añadido, se nos mostrará la pantalla principal de la gestión de recursos.

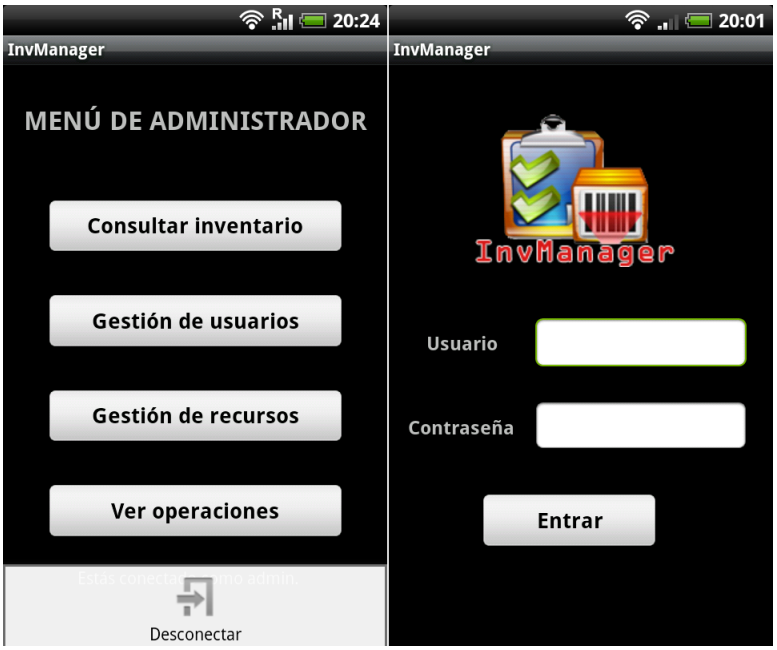


Ilustración 49: Pantallas de menú de administrador con botón de desconectar y pantalla principal

Por último, en la parte izquierda de la ilustración 49 vemos el menú principal de administrador con la opción de desconectar de la aplicación. A la derecha, pantalla principal de la aplicación, que se nos mostrará al desconectar.